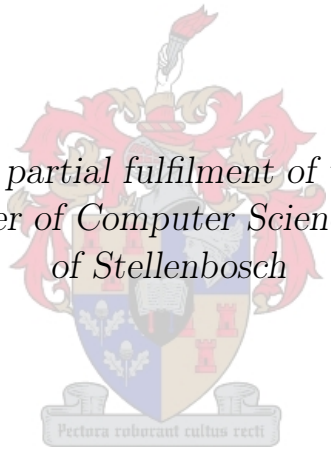
Ontology Comprehension

by

Johann Rath Bergh



Thesis presented in partial fulfilment of the requirements for the degree of Master of Computer Science at the University of Stellenbosch

Division of Computer Science
Stellenbosch University
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Prof. A. Gerber, Prof. L. van Zijl

December 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
J.R. Bergh

Date: .....

Abstract

Ontology Comprehension

J.R. Bergh

Division of Computer Science
Stellenbosch University
Private Bag X1, 7602 Matieland, South Africa

Thesis: MSc (Computer Science)

December 2010

Ontologies are conceptual models of a domain of discourse and are used in a number of applications to model a field of knowledge. For example, SNOMED, an ontology of medical terminology, is widely used among medical professionals. Commercial ontologies, such as SNOMED, can have hundreds of thousands of concepts. People who want to use these ontologies need an understanding thereof, but the sheer magnitude of these ontologies hampers comprehension. It was within this context that the need arose for software tools that facilitate the understanding of ontologies. Given this background, our aim is to investigate a new area within the field of ontologies, namely, ontology comprehension. We make a contribution to it by developing an ontology comprehension framework and writing a software tool of our own. This software tool, PathViz, helps users to understand how different concepts in an ontology are related to each other and what effect entailments have on the way concepts in an ontology relate to each other. The ontology comprehension framework, PathViz and the reasoning measurement instruments were found useful for ontology comprehension by participants at an ontology workshop.

Opsomming

Ontologie-begrip

J.R. Bergh

Afdeling Rekenaarwetenskap
Universiteit van Stellenbosch
Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MSc (Rekenaarwetenskap)

Desember 2010

Ontologieë is konseptuele modelle van 'n domein en word in verskeie toepassings gebruik om 'n kennisveld te modelleer. SNOMED is 'n voorbeeld van 'n ontologie van mediese terme wat baie gebruik word deur die mediese beroepslui. Kommersiële ontologieë, soos SNOMED, kan bestaan uit duisende konsepte. Dit is belangrik om hierdie ontologieë wat gebruik word te verstaan, maar die enorme omvang van hierdie ontologieë belemmer die verstaanproses. In hierdie konteks het die behoefte ontstaan vir programmatuur wat die verstaanproses van ontologieë vergemaklik. Met hierdie agtergrond inaggenome, is dit ons doel om 'n nuwe area in die veld van ontologieë te ondersoek, naamlik, Ontologie-begrip. Ons maak 'n bydra tot hierdie veld deur 'n raamwerk vir ontologie-begrip te ontwikkel en programmatuur van ons eie te skryf. Hierdie programmatuur, PathViz, help gebruikers om te verstaan hoe verskillende konsepte in 'n ontologie aan mekaar verwant is. Verder help dit gebruikers om te verstaan watter invloed afleidings uit die ontologieë het op konsepverwantskappe. Deelnemers aan 'n ontologie-werkswinkel het gevind dat die raamwerk vir ontologie-begrip, PathViz en die instrumente wat die invloed van die ontologie-redeneerder meet, ontologie-begrip bevorder.

Acknowledgements

- The KRR Research Group in Meraka at the CSIR for funding for this research.
- My supervisor (Prof. A. Gerber) and co-supervisor (Prof. L. van Zijl) for their hard work and support.

Dedication

Aan my ouers

Contents

One

Introduction

1.1 Motivation

Ontologies are conceptual models of a domain of discourse [46]. Ontologies are used in a number of applications to model a field of knowledge. For example, SNOMED [8], an ontology of medical terminology, is widely used by medical professionals. Commercial ontologies, such as SNOMED, can have millions of concepts. Users who want to use these ontologies need an understanding of it, but the sheer magnitude of these ontologies hampers comprehension. It was within this context that the need arose for software tools that facilitate the understanding of ontologies.

Currently, there are many software tools that visualise certain aspects of ontologies [37]. For example, OWLViz [2, 37] is a two-dimensional Protégé plug-in that visualises the asserted and inferred class hierarchies in an OWL (Web Ontology language) ontology. ClusterMap [23], on the other hand, is a two dimensional software tool that focuses on the visualisation of individuals in an ontology. However, there are not many software tools that focus on facilitating the understanding of an ontology. SuperModel [16, 17] is one such software tool. SuperModel builds models of an ontology, that serve as practical examples to give users an idea of how an ontology can be used. Users can manipulate these models and test the consequent satisfiability of the models.

Given this background, our aim is to formalise a framework (Definition 5.1 in Chapter 5) wherein tools and techniques can be developed to facilitate the understanding of an ontology. In addition, we also want to seek new ways to facilitate understanding within such a formal framework. We do this by designing a software artefact, PathViz, that is a Protégé plug-in. PathViz implements a technique that aids users to understand an ontology. This technique focuses on the way concepts in an ontology are related to each other (Definition 5.3 in Chapter 5). An analysis of this software artefact leads to an investigation into the effect that entailments have on concept relationships in an ontology. Finally, we formalise measurement instruments that give an indication of the effect that entailments have on concept

relationships in an ontology. Definition 7.1 and Definition 7.2 in Chapter 7 describe these measurement instruments.

1.2 Background

In the information systems context, ontologies refer to a conceptual model of a domain of discourse. The rules that capture knowledge about an ontology can be written in a mathematical language. When these rules are written as a set of mathematical statements, it is called knowledge representation formalisms (KRFs) [46]. Description Logics (DL) is a prominent KRF that is used to represent an ontology.

A KRF or a combination of KRFs is the foundation of an ontology language. There already exist several mark-up languages such as XML that can persist data. Ontology languages take the principle of persisting defined data from XML, but enrich it by allowing the storage of more complex information such as detailed relationships between concepts [33]. OWL is an ontology language that has become a W3C standard [35]. When creating and editing an ontology in an ontology editor, such as Protégé, the ontology is persisted to a *.owl* file.

Embedded in ontologies are knowledge representation techniques that enable reasoning. This means that ontologies capture the knowledge of a particular domain as computational artefacts [46]. Computational artefacts are parts of the domain knowledge and they are captured in such a way in an ontology that reasoning can take place. Reasoning refers to the process of deriving implicit knowledge from explicit knowledge in an ontology. Ontology editors, such as Protégé, have different implementations of reasoners. Two widely used reasoners in Protégé are Fact++ and Pellet [46].

Ontologies that are used in practice can become too large and complex to understand. However, quick comprehension is beneficial for the users of ontologies. In the business world, quick comprehension can be crucial in obtaining a profit. Researching comprehension can be rather difficult, because it is not easily measurable. In this research, we aim to construct artefacts that aid the comprehension of an ontology.

1.3 Problem statement and purpose of this study

Several research threads within ontologies have been classified and ordered to some extent to give the community a clearer context and understanding. For example, Katefori *et al.* [37] classified ontology visualisation software. Although some authors mention the term *ontology comprehension* in different contexts [21, 25, 38], there is no formal framework that defines *ontology comprehension*. We argue that the

development of an ontology comprehension framework will be useful, as was the case with the ontology visualisation classification of Katefori *et al.*

Within an ontology comprehension framework, current tools and techniques relating to ontology comprehension can be classified. One such software tool is SuperModel [16, 17]. An ontology comprehension framework also enables the development of new tools and techniques that aid ontology comprehension. Here, any novel idea that helps users to understand ontologies, will qualify. This will then address a gap or shortcoming and thus contribute to the scientific body of knowledge. In this regard, we aim to make a contribution by addressing two aspects. Firstly, we propose that an ontology can be better understood if we highlight in detail the relationships amongst concepts in an ontology. An implementation of such an idea can take the form of an artefact that uses path visualisation techniques. Secondly, there is currently no way to measure the effect that entailments have on concept relationships. Katefori *et al.* [37] remark that the representation of reasoning (or the effect of a reasoner on an ontology) in ontology visualisation is not satisfactory:

> *A very important issue related to ontologies, which are mainly knowledge representations, is that of reasoning. An ontology is more than a simple graph, it is a structure with rich semantics and the ability to use logic operations on it so as to reach conclusions and produce new information. The issue of coupling visualisation and reasoning has not yet been sufficiently treated in existing literature and very few methods support it.*

The aim is to find measurement instruments that can be used to measure the effect that entailments have on concept relationships.

In summary, our investigation focuses on the implementation of path visualisation techniques in order to better understand concept relationships and enhance ontology comprehension. In order to conduct this kind of research in a focused way, we compiled several research questions.

1.4 Research questions

The following research questions were compiled:

	Description of research question
Main research question (Q_0)	How can the use of path visualisation techniques applied to subsumption and existential relationships between concepts in an ontology, enhance ontology comprehension within an ontology comprehension framework?
Sub research question 1 (Q_1)	How can we construct an ontology comprehension framework wherein we can classify the approaches related to ontology comprehension?
Sub research question 2 (Q_2)	How can we apply path visualisation techniques to comprehend subsumption and existential relationships between concepts in an ontology?
Sub research question 3 (Q_3)	How can path visualisation techniques be used to comprehend the effect of the reasoner in a formal ontology?

Table 1.1: Research questions

1.5 Scope and context of the study

This study includes an investigation into existing ontology visualisation techniques. As a point of departure, this investigation shows how visualisation techniques in general aid understanding.

An investigation into ontology reasoning is done. Here, the focus is specifically on showing why it is not always easy to understand the effect of an ontology reasoner.

The construction of a software artefact, PathViz, was implemented in the Protégé ontology editor as a plug-in. The Fact++ and Pellet ontology reasoners were employed as they are widely used in Protégé. Protégé uses OWL as ontology language with Description logics (DL) as the underlying KRF. In the context of the PathViz implementation, discussions in consequent chapters focus on OWL as ontology language and DL as the underlying KRF.

Investigations into concept relationships in an ontology focus on existential and subsumption relationships. Furthermore, we argue that existential relationships can be entailed from minimum and exact cardinality relationships.

Measurement instruments are developed to give an indication of the effect that entailments have on concept relationships. These measurement instruments make

calculations based on the results obtained of an ontology reasoner and existential and subsumption relationships in an ontology.

1.6 Limitations of scope

In this research, the following limitations apply:

- As far as KRFs are concerned, the focus is on DL as a KRF and OWL as an ontology language. Other KRFs and ontology languages are not considered. This is mainly due to the fact that the Protégé ontology editor uses OWL as ontology language.
- In the consideration of concept relationships in an ontology, the focus is on existential and subsumption relationships. Universal relationships are not considered at this point in time, mainly due to time constraints.

1.7 Research method

The chosen method to address the research questions as stated above is *design research*. The plan is to construct artefacts and evaluate these artefacts to address the research questions. Firstly, an ontology comprehension framework is constructed by means of a literature analysis. Secondly, an existing software tool, SuperModel, is classified in this framework and a new software tool, PathViz, was developed within this framework. The focus of PathViz was to facilitate the understanding of concept relationships in an ontology by means of path visualisation techniques. Finally, an analysis of PathViz motivates the development of measurement instruments that give an indication of the influence of entailments on concept relationships in an ontology.

1.8 Outline of the study

Chapter 2 gives an overview of ontologies. Chapter 3 explains the significance of visualisation in the process of understanding within the context of ontologies. Chapter 4 discusses reasoning in ontologies and why it is difficult to understand the effect of an ontology reasoner. Chapter 5 describes an ontology comprehension framework and addresses Q_1. Chapter 6 explains what PathViz is and how it was built. PathViz is an implementation that proposes to answer Q_2. From the PathViz implementation we derive measurement instruments that help us to understand the effect that entailments have on concept relationships. We give more details on these measurement instruments in Chapter 7 that will aim to answer Q_3. We evaluate PathViz in Chapter 8, before we conclude in Chapter 9. Appendix A gives

an overview of Protégé and plug-in development in Protégé. Appendix B provides technical details on PathViz, a Protégé plug-in. This outline is depicted graphically in Figure 1.1.

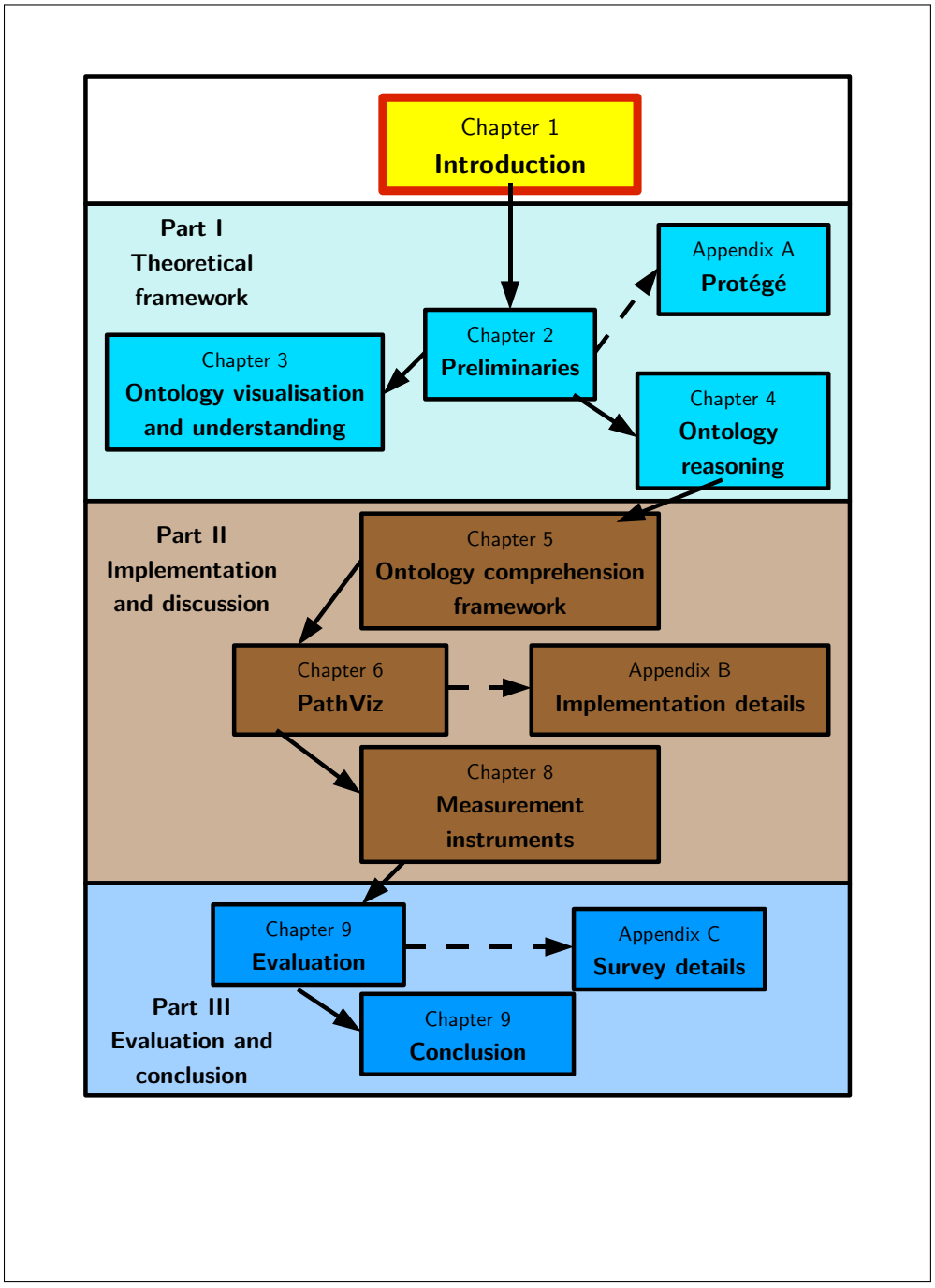


Figure 1.1: Chapter layout

Two

Preliminaries

2.1 Introduction

This chapter discusses fundamental principles in ontologies that will feature throughout this thesis. Section 2.2 describes what an ontology is. A discussion on description logic (DL) can be found in Section 2.3. OWL is an ontology language and is discussed in Section 2.4. Section 2.5 elaborates on software tools that are useful for creating and editing ontologies.

2.2 Ontology

The word *ontology* was originally used to refer to the study of the nature of being, existence or reality in general [29, 43]. Ontologies have a different meaning in a computer science context. We will be studying ontologies from the computer science perspective. Here, ontologies refer to a conceptual model of a domain of discourse.

Gruber [27] describes an ontology as an explicit specification of a conceptualization. Pretorius [43] describes an ontology (in the context of computer and information science) as a designed artefact that formally represents agreed semantics in a computer resource.

Ontologies consist of concepts, relations and instances (known as the ontological vocabulary) [46]. Conceptual models represent a domain of discourse. *Concepts* represent classes of objects relating to the domain of interest [46]. A physical model is a specific (concrete) implementation of a conceptual model. *Instances* represent concrete objects (specific implementation of a concept) in the domain of interest. *Relations* semantically connect concepts to each other [46].

Figure 2.1 illustrates how the elements of the ontological vocabulary relate to each other. Here, `Teacher` and `Student` are both concepts. `Prof Smith` is a physical instance of `Teacher`. `John Brown` is a physical instance of `Student`. The relation `teaches` links two concepts (`Student` and `Teacher`) as well as two instances

(`Prof Smith` and `John Brown`)

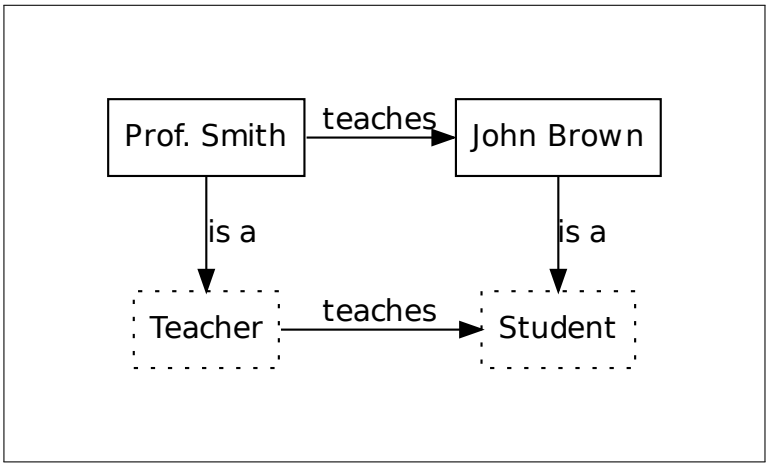


Figure 2.1: Ontological vocabulary example

2.3 Description Logics

2.3.1 Background

The rules that capture knowledge about an ontology can be written in a mathematical language. When these rules are written as a set of mathematical statements, it is called knowledge representation formalisms (KRFs) [46]. A prominent KRF paradigm that is used in the world of ontologies is *Description Logics* (DL) [46]. In this section, the aim is to highlight the most important aspects of DLs.

DLs use the ontological vocabulary as basic building blocks to represent knowledge. The initial assumption is that there is a set of *concepts*, a set of *relations* and a set of *instances*. By combining elements of these sets with each other, complex concept expressions can be formed. An example (taken from [14]) illustrates DLs:

$$Human \sqcap \neg Female \sqcap \exists married.Doctor \sqcap (\geq 5 hasChild) \sqcap \forall hasChild.Professor$$

The meaning of the given example is

> *A human that is not a female and that is married to a doctor and has at least five children, all of whom are professors.*

The given example can be seen as a *formula* (a logical statement in mathematical terms). A knowledge base consists of several of these formulas. In the example, several boolean constructors are employed: *conjunction* ($\sqcap$), *negation* ($\neg$), *universal restriction* ($\forall r.C$), *existential restriction* ($\exists r.C$), *number restriction* ($>$, $<$, $=$).

Formulas that only employ these five boolean constructors are also referred to as *description formalisms* [14]. We briefly elaborate on each of these contructors:

- Conjunction can be seen as set intersection.
- Negation is interpreted as set complement.
- Universal restriction is always written in the form $\forall r.C$ where r refers to some relation and C is a class (concept). The expression $\forall r.C$ is a class in its own right. For example, $\forall hasChild.Professor$ is a class that contains everybody that has children such that all of them are professors.
- The existential restriction is always written in the form $\exists r.C$ where r refers to some relation and C is a class. The expression $\exists r.C$ is also a class in its own right and the example $\exists married.Doctor$ would group everybody that has at least one marriage with a doctor in this class.
- Number restriction is written in the form $\geq nR$ where n is some integer value and R refers to some relation. For example, $\geq 5hasChild$ would equate to everybody that has at least 5 children.

Apart from description formalisms, there are two other formalisms in DL, namely *terminological* and *assertional formalisms* [14]. Terminological formalisms (also referred to as *TBox* statements) are formulas that represent *concept inclusions* ($C \sqsubseteq D$) or *concept equivalences* ($C \equiv D$). Assertional formalisms (also referred to as *ABox* statements) state properties about particular individuals (instances). *Concept assertions* are written in the form $C(a)$ and *role assertions* in the form $r(a, b)$ [46]. Table 2.1 gives a summary with examples of the different formalisms used in DLs.

Interpretation is the next important concept that relates to DLs. The formulas in an interpretation are assembled from a set of relations (roles), concepts and instances (the same sets that were used to assemble the DL knowledge base). During the reasoning process in ontologies, the formulas in the interpretation are compared with those in the DL knowledge base. If the formulas in the interpretation do not contradict those in the DL knowledge base, then the interpretation is called a *model* of the DL knowledge base (meaning that the interpretation is a correct reflection of the truths in the DL knowledge base). Certain lines of research focus on correct interpretations and reasoning methods, with the danger of neglecting the correct construction of the DL knowledge base [28]. Advanced interpretations and reasoning methods are not advantageous, if it is applied on an incorrect model of reality. Guarino [28] emphasizes the importance of well-constructed knowledge bases in order to obtain correct results from reasoning methods.

In Figure 2.2 we summarise how DLs and knowledge bases relate to each other. Here, DLs consist of three different types of formalisms. Formulas are constructed

Formalism	Example	Meaning
Description $(\sqcap, \neg, \forall r.C, \exists r.C,$ $\geq nR)$	$\neg Female \sqcap \exists married.Doctor$	A non-female that is married to a doctor
Terminological $(C \sqsubseteq D, C \equiv D)$	$Employee \equiv \exists employedAt.Company$ $Male \sqsubseteq Human$	Employees are exactly those people employed at some company Males are Humans
Assertional $(C(a), r(a,b))$	$IsHappy(EmployeeX)$ $WorksFor(EmployeeX, CompanyY)$	EmployeeX is happy EmployeeX works for CompanyY

Table 2.1: Summary of formalisms in DLs

within the context of these formalisms, and the knowledge base contains many formulas. Interpretations consist of a set of formulas. Valid interpretations are models of the knowledge base.

2.3.2 Satisfiability, consistency, validity and coherency

The terms *satisfiability, consistency, validity* and *coherency* are ubiquitous terms in DLs and therefore we discuss it in a separate section. Several authors (for example [34, 35, 46]) have discussed these terms.

A class in an ontology is said to be *satisfiable* if there exists a circumstance wherein the class can have an instance. Similarly, a class in an ontology is unsatisfiable if there is no circumstance wherein the class can have an instance.

A set of fromulae is said to be *consistent* if it is possible for all of them to be true. An ontology is said to be consistent when the associated DL knowledge base is satisfiable. In other words, there exists a circumstance wherein all the classes and formulae in the DL knowledge base can be true.

A formula in an ontology is *valid* if and only if it is true under every possible interpretation.

An ontology is coherent when all its *named* classes are satisfiable concepts.

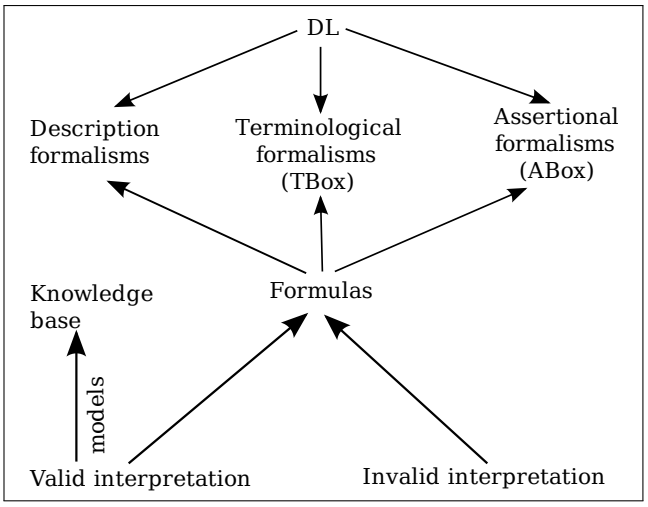


Figure 2.2: DLs and knowledge bases (Summary of Section 2.3.1)

2.3.3 Varieties of DLs

There are many varieties (different types) of DLs. Baader *et al.* [15] call these varieties of DLs *description languages*. Furthermore, they state that description languages are distinguished by the constructors that they provide. The most basic description language is refered to as $\mathcal{AL}$ (Attributive Language) [15]. In $\mathcal{AL}$, the following is permissible [15]:

- Atomic concept (A)
- Universal concept ($\top$)
- Bottom concept ($\bot$)
- Atomic negation ($\neg A$)
- Intersection ($C \sqcap D$)
- Value restriction ($\forall R.C$)
- Limited existential quantification ($\exists R.\top$)

The $\mathcal{AL}$ language can be extended by adding additional constructors [15]. Description languages that are formed in this way are referred to as the family of $\mathcal{AL}$-languages [15]. For example, *full existential quantification* ($\exists R.C$) is represented by the letter $\mathcal{E}$. Therefore, the language $\mathcal{ALE}$ has the same expressivity as basic $\mathcal{AL}$ plus full existential quantification.

The letter $\mathcal{S}$ was introduced for description languages that extend $\mathcal{ALC}$ by transitive roles [13]. Consequently, there also exist even more expressive description

languages such as $\mathcal{SIN}$, $\mathcal{SHIF}$, $\mathcal{SHIQ}$ and $\mathcal{SHOIN}$ [13, 31, 32]. For example, $\mathcal{SIN}$ refers to the description language that has the same expressivity as $\mathcal{ALC}$ plus transitive roles, inverse properties and cardinality restrictions.

The meaning of each of the letters in the name of a description language is explained in various literature sources (for example [13, 15]) and we summarise it in Table 2.2.

Letter	**Meaning**
$\mathcal{C}$	Complex concept negation
$\mathcal{E}$	Full existential qualification
$(\mathcal{D})$	Data type properties; data values or data types
$\mathcal{F}$	Functional properties
$\mathcal{H}$	Role hierarchies
$\mathcal{I}$	Inverse properties
$\mathcal{N}$	Cardinality restrictions
$\mathcal{O}$	Nominals
$\mathcal{Q}$	Qualified cardinality restrictions
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness
$\mathcal{S}$	Abbreviation for $\mathcal{ALE}$ with transitive roles
$\mathcal{U}$	Concept union

Table 2.2: Expressivity in description languages

Three other description languages that do not conform to the above-mentioned convention are $\mathcal{FL}^-$, $\mathcal{FL}_0$ and the $\mathcal{EL}$-family [13, 15]. The $\mathcal{FL}^-$ description language is a sub-language of $\mathcal{AL}$, obtained by disallowing atomic negation [15]. The $\mathcal{FL}_0$ description language is a sub-language of $\mathcal{FL}^-$, obtained by disallowing limited existential quantification [15]. The $\mathcal{EL}$ description language allows only two constructors, namely intersection of concepts and existential quantification [13]. Extentions of $\mathcal{EL}$ can be obtained by adding the appropriate symbols from Table 2.2. For example, $\mathcal{ELU}$ has the same expressivity as $\mathcal{EL}$ plus concept union.

Finally, some custom applications, most notably in the medical field, make use of their own custom description language to suite the needs of their application [22]. For example, SNOMED RT and CT [8] makes use of a description language that allows for conjunction, existential quantification and the top-concept [22]. GALEN, a model of medical concepts, uses a description language called GRAIL (GALEN Concept Representation Language) [22]. GRAIL extends the description language used in SNOMED by allowing additional role constructors such as role chaining [22] which refers to a list of roles linked to each other with role composition ($R \circ \ldots \circ R$).

2.4 OWL

2.4.1 Introduction

A KRF or a combination of KRFs is the foundation of an ontology language. Ontology languages allow users to write explicit, formal conceptualizations of domain models [12].

OWL is a widely used ontology language and is a W3C standard [35]. RDF was the ontology language that preceded OWL, but it was not expressive enough to be used in the context of the semantic web [12]. This shortcoming resulted in the development of OWL as an ontology language. Although the initial implementation of OWL was largely successful, users of ontologies also indicated that there were limitations [26] such as the absence of qualified cardinality restrictions. To address these limitations a new W3C working group for OWL was formed and they implemented a versioning system for OWL to indicate the progress in the development of OWL as an ontology language. They refer to the initial version of OWL as *OWL 1* and the current version of OWL as *OWL 2* [26].

2.4.2 OWL 1

OWL 1 has three different versions (also referred to as the species of OWL 1) [12], that differ in their levels of expressivity. *OWL Full* is the most expressive ontology language that is fully backward compatible with RDF. *OWL DL* is a sub-language of OWL Full that restricts the way in which constructors from OWL and RDF can be used. *OWL Lite* further limits OWL DL to a subset of the language constructors.

The choice of OWL 1 species, will depend on the goal the user wants to achieve. OWL Full is the most expressive, but can become undecidable [26] and therefore is limited in the reasoning support it can provide. OWL Lite on the other hand is restricted in expressivity, but provides extensive reasoner support. The ontology developer will have to consider the trade-off between expressivity and reasoning capabilities for the task at hand when choosing the species of OWL 1 to use.

2.4.3 OWL 2

Ontology users identified certain limitations in OWL 1 [26]. OWL 2 was developed to address these limitations. Grau *et al.* elaborate on these limitations and how they were addressed in OWL 2. The new features in OWL 2 are also summarised in [1]:

- keys (unique identifiers)
- property chains

- richer data types and data ranges
- qualified cardinality restrictions
- asymmetric, reflexive and disjoint properties
- enhanced annotation capabilities

Like species in OWL 1, OWL 2 has a similar sub-division of the language. These sub-divisions or sub-languages are called *profiles* [1, 26]. Grua *et al.* [26] refer to profiles as trimmed down versions of the OWL 2 language that trade some expressive power for the efficiency of reasoning.

The profiles specified in OWL 2 are *OWL 2 EL, OWL 2 QL, OWL 2 RL.* In [1] OWL 2 EL is described as a sub-language that is suitable for large ontologies where expressive power can be traded for performance guarantees. OWL 2 QL on the other hand is suitable for lightweight ontologies that are used to organise large numbers of individuals. Grau *et al.* [26] describe OWL 2 RL as a sub-language that has been designed so that several reasoning tasks can be implemented as a set of rules in a forward-chaining rule system.

2.4.4 Conclusion

OWL is a widely used ontology language and a W3C standard. A versioning system was implemented for OWL to indicate the progress in the development of OWL as an ontology language. OWL 2 is the latest version of the OWL ontology language.

OWL as an ontology language is linked to description languages (described in Section 2.3.3) in the sense that sub-languages of OWL correspond in their expressivity to a description language [26, 32]. For example, OWL 1 DL corresponds to $\mathcal{SHOIN}(\mathcal{D})$ and OWL 1 Lite corresponds to $\mathcal{SHIF}(\mathcal{D})$ [26, 32].

2.5 Software editors for ontologies

SNOMED [8], an ontology of medical terminology, is widely used by medical professionals. Commercial ontologies, such as SNOMED, can have hundreds of thousands of concepts. Creating large ontologies and performing reasoning on them are time-consuming tasks. Within this context, the need arose for software editors that aid humans in creating large ontologies and performing reasoning tasks on them. Volz [46] describes a list of software editors for building and using ontologies: Swoop, Ontostudio and Protégé.

Kalyanpur *et al.* [36] describe Swoop as a web ontology browser and editor that is specifically tailored for OWL ontologies. Swoop allows for hypertextual navigation of ontologies. Another distinct feature of Swoop is the easy cross-linking of entities in different ontologies [36].

Ontostudio is a commercial graphical ontology editing tool that provides support for OWL ontologies [46]. Ontostudio also provides support for database schemas and WebServices [46].

Protégé is a widely used ontology editor and we focus on it the context of this study. An understanding of the main features of Protégé (important from a user point of view) as well as its software architecture (important from a software developer point of view) is important within the context of this work and the reader can find such a discussion in Appendix A. Discussions in Chapter 6 will describe a software plug-in that was developed for Protégé and foreknowledge of the Protégé architecture is assumed.

2.6 Conclusion

This chapter described some fundamental aspects of ontologies that will be important for the rest of this study. Additionally, this chapter was the first chapter in the theoretical framework of this study. The next two chapters, Chapter 3 and Chapter 4, are the remaining chapters in the theoretical framework.

Three

Visualisation and understanding

3.1 Introduction

The purpose of this chapter is to explain the role of visualisation and understanding within the context of ontology comprehension. Section 3.2 starts by explaining why visualisation in general aids understanding. In Section 3.3 we continue by giving an overview of ontology visualisation literature that is relevant within the context of ontology comprehension. In Section 3.4 we confirm that visualising ontologies is useful for understanding, but we argue that the visualisation of a specific ontology understanding technique further enhances understanding. We do this by giving an overview of the work Bauer [16, 17] did in model exploration and ontologies. Section 3.5 concludes.

3.2 Significance of visualisation

Much research has been done within the field of ontology visualisation (for example [11, 19, 37]). We proceed to argue that ontology visualisation techniques are the only tools currently available that facilitate the understanding of ontologies. It is well-known that visualisation techniques, in general, facilitate better understanding of complex systems [18, 48].

We show, by means of two examples, that visualisation techniques aid understanding. Example 3.1 describes the structure of a fictitious company. Example 3.2 describes the provincial divisions of South Africa. Both examples are accompanied by corresponding images.

Example 3.1 *ComprehensionTech is a consulting firm that helps businesses to clean up their information and understand their business processes. Andrew is the founder and CEO of ComprehensionTech. Bob is the CFO of the company and serves on the board of directors with Andrew. ComprehensionTech have an IT and marketing department. Candice is the manager of the IT department. The IT*

department has three programmers and two business analysts. Don, Edward and Frank are programmers. Gale and Howard are business analysts. The marketing department is managed by Ingrid. Jennifer is her secretary and Kaleb and Laurence are general administration clerks.

In this example, the reader will have to study the information intensively before obtaining a complete understanding of the company structure. On the other hand, a quick glance at the image (Figure 3.1) that accompanies the example gives the reader an immediate grasp of the company structure. Ambiguities in the wording of the example are also eliminated in the image. For example, from the final sentence it is unclear whether `Kaleb` *and* `Laurence` *are general administration clerks for the company as a whole, or whether they only work for the marketing department. In the image, however, it is clear that they only work for the marketing department.* □

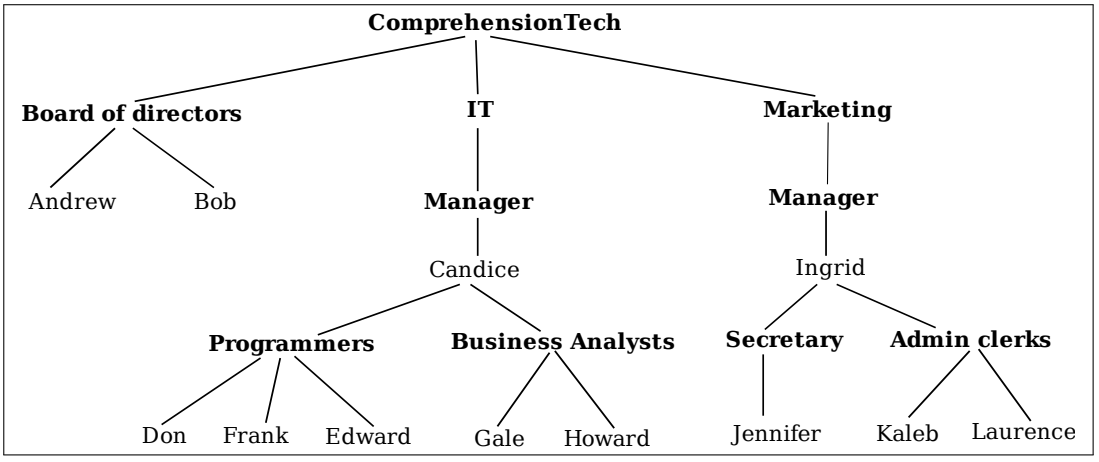


Figure 3.1: ComprehensionTech company structure

Example 3.2 *In a second example there is an image (Figure 3.2) of South Africa and a corresponding description next to it. The image is not only easier to understand, but also gives additional information by depicting the physical shape of South Africa. In addition we observe that Lesotho is a country that is landlocked within the borders of South Africa.* □

3.3 Ontology visualisation

Several software tools have been written to visualise ontologies. Katefori *et al.* [37] elaborate on the implementation of such software tools and the visualisation theory that was used. In this section, some of these software tools and their visualisation

South Africa is a country that is situated on the southern tip of Africa. South Africa has nine provinces: Western Cape, Eastern Cape, Northern Cape, Free State, Natal, Mpumalanga, Gauteng, North West and Limpopo. The neighbouring countries of South Africa are Lesotho, Mozambique, Swaziland, Namibia and Botswana.

Figure 3.2: South Africa (taken from [7]) and corresponding description

theory will be discussed. Ideally, we want to present a framework to describe ontology visualisation tools. Two fundamental questions within ontology visualisation are:

- *What* do we want to visualise?
- *How* do we want to visualise it?

What needs to be visualised, are the elements contained in ontologies. These elements are the concepts, relations and instances (ontological vocabulary) [46] in an ontology. The aim is to evaluate to what extent currently available software tools address the visualisation of these aspects.

Secondly, the focus falls on *how* to visualise ontologies. Katefori *et al.* [37] proposed a framework wherein they evaluated the visualisation methods of the ontology visualisation software tools. Within this framework, visualisation methods were grouped into categories. For example, a category *Zoomable Visualisations* would describe the ontology visualisation software tools with methods that allow users to interactively increase or decrease the granularity of their view. In this approach overlapping can occur between categories and therefore duplication is unavoidable. In other words, a visualisation method can be used in more than one category.

We propose a different framework for categorising the visualisation of ontologies. This framework is based on three principles: *visualisation methods, visualisation properties* and *evaluation criteria.*

- The notion of *visualisation methods* is retained from Katefori *et al.* [37], noting that it refers to the layout of the ontological vocabulary on the available

screen space. An example of a visualisation method is the *hyperbolic tree* (Figure 3.3). This method optimises use of screen space to display information in a tree-like structure.

- *Visualisation properties* refer to characteristics of specific elements within a visualisation. For example, the colour and size of an element within a visualisation are visualisation properties. Other visualisation properties of note are shape and opacity. Visualisation properties facilitate the differentiation between the elements in a visualisation [44].

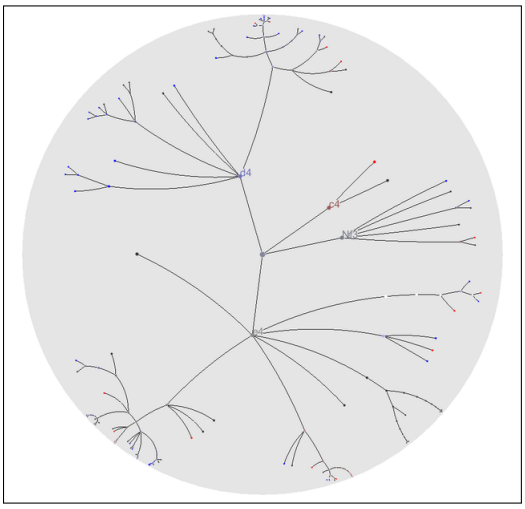


Figure 3.3: Example of a hyperbolic tree (taken from [9])

Visualisation methods and properties can be seen as tools that enable visualisations.

- *Evaluation criteria*, the final principle in our proposed framework, can be used to judge whether visualisation methods and properties used in a visualisation are successful as a whole. Preece *et al.* [42] give examples of design and usability principles that can be used to evaluate software tools.

A summary of our proposed framework is displayed in Figure 3.4.

At this point, our discussion continues by using two principles (visualisation methods and visualisation properties) in our proposed framework to describe existing work within the field of ontology visualisation. We do not consider *evaluation criteria* at this point in time, because our aim is not to judge whether the visualisations in these software tools were successful on a technical level (for example if the right colour combinations were used). Our aim is to discuss *what* is visualised in ontologies and *how* it is visualised.

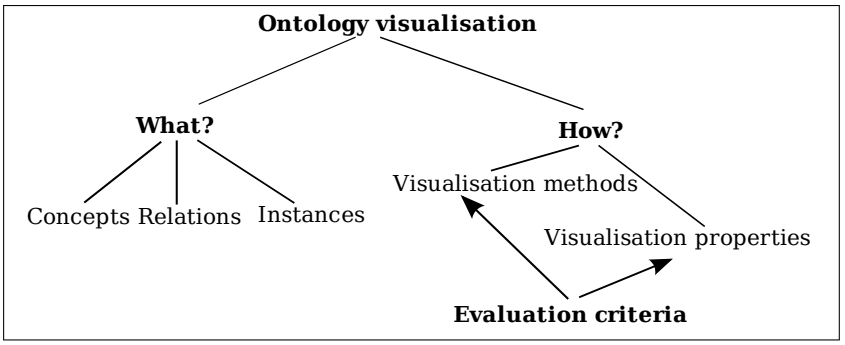


Figure 3.4: Ontology visualisation framework

Since Katefori *et al.* [37] have completed a thorough investigation into current ontology visualisation tools, we focus on work relevant within the context of ontology comprehension. Some of the current ontology visualisation tools, for example [2, 11], focus solely on visualising the concepts, roles and relationships in an ontology (to be referred to as *trivial visualisations*). Other ontology visualisation tools, for example [24, 39], are more *intelligent* in the sense that they visualise more than merely the concepts, roles and relationships in an ontology (to be referred to as *complex visualisations*). In our approach we consider a representative selection of literature, where this selection comprises both trivial and complex visualisations. We will progressively discuss these ontology visualisation tools, where we start with the tools that use the most trivial visualisations and end with the tools using the most complex visualisations. The final tool in our selection is *SuperModel* [16, 17]. This is an ontology visualisation tool that aids the user in understanding ontologies. SuperModel uses model exploration techniques to achieve its goal. In the next section, we elaborate on this notion by explaining that we do not regard SuperModel merely as a ontology visualisation tool, but as a visualisation tool that specializes in facilitating the understanding of ontologies. Table 3.1 displays the ontology visualisation tools for this discussion. Those tools marked in a lighter shade of grey use more trivial visualisations, while tools using more complex visualisations are progressively marked in a darker shade of grey.

OWLViz	TGVizTab	OntoSphere	Swoop	ClusterMap	VantagePoint	SuperModel

Table 3.1: Selection of ontology visualisation tools for evaluation

At this point, it is instructive to note that we do not regard the software tools with trivial visualisations as inferior to those with more complex visualisations. These tools merely differ in purpose and functionality.

OWLViz [2, 37] is a two-dimensional Protégé plug-in that visualises the class hierarchies in an OWL ontology. The visualisations of these class hierarchies can be navigated. Role relationships in the ontology are not included in the visualisations. An advantage of OWLViz is that both the asserted and inferred class hierarchies are visualised. These two visualisations can be compared to each other. OWLViz uses a tree-like layout structure to position the elements of the visualisation on the screen. Two visualisation properties of note in OWLViz are colour and shape. For example, concepts in the class hierarchies are displayed in an ellipse with a yellow background and equivalent concepts in the class hierarchy are displayed in an orange background. Inconsistent concepts (concepts that could not be classified) are highlighted in red. The main focus of OWLViz is to visualise the ontological vocabulary within the context of the taxonomy. Figure 3.5 depicts OWLViz.

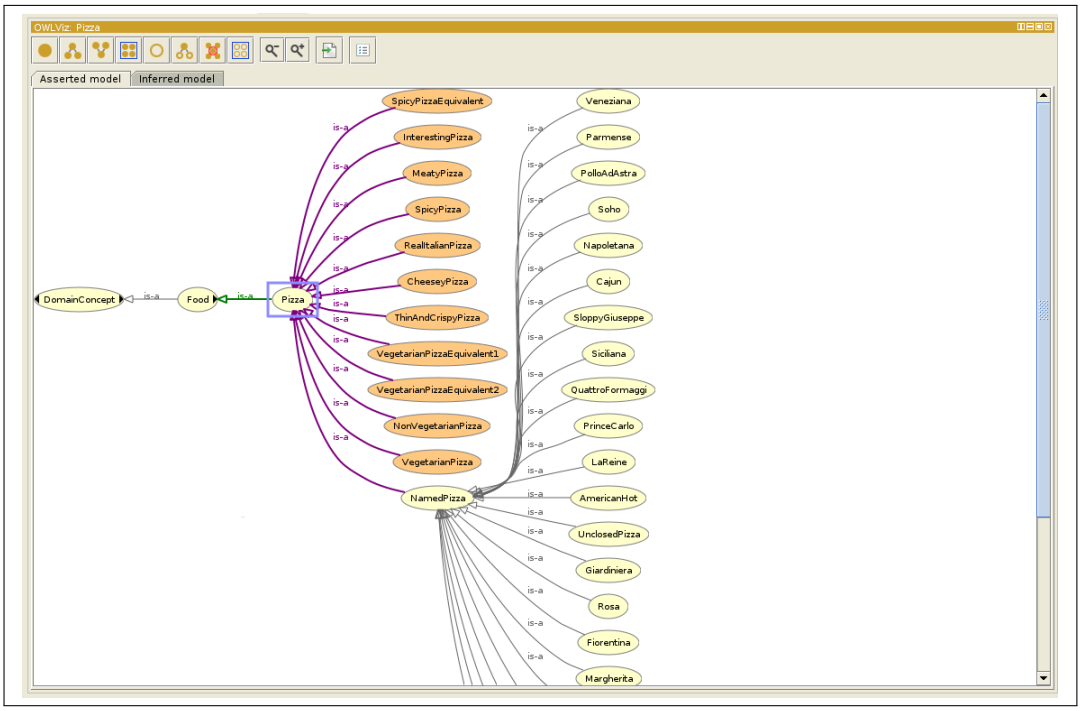


Figure 3.5: OWLViz (Protégé screen-shot)

TGVizTab (Figure 3.6) is another two-dimensional Protégé plug-in that visualises ontologies. Ontologies are visualised by using directed networks of graphs that depict the classes, instances and relations in an ontology [11]. The visualisations in TGVizTab are more comprehensive than in OWLViz (the second entry in Table 3.1). Firstly, role relations are included in the visualised graphs, while OWLViz only visualises hierarchical relationships. Secondly, TGVizTab visualises instances created within the ontology. TGVizTab makes use of a *spring-layout*

technique as a visualisation method. This method works on the principle of unconnected graph nodes repelling each other, and nodes connected to each other with graph edges, attracting each other. The result is that semantically similar nodes are placed closer to each other on the screen. TGVizTab uses colour as a visualisation property to distinguish between classes and instances in the visualisation.

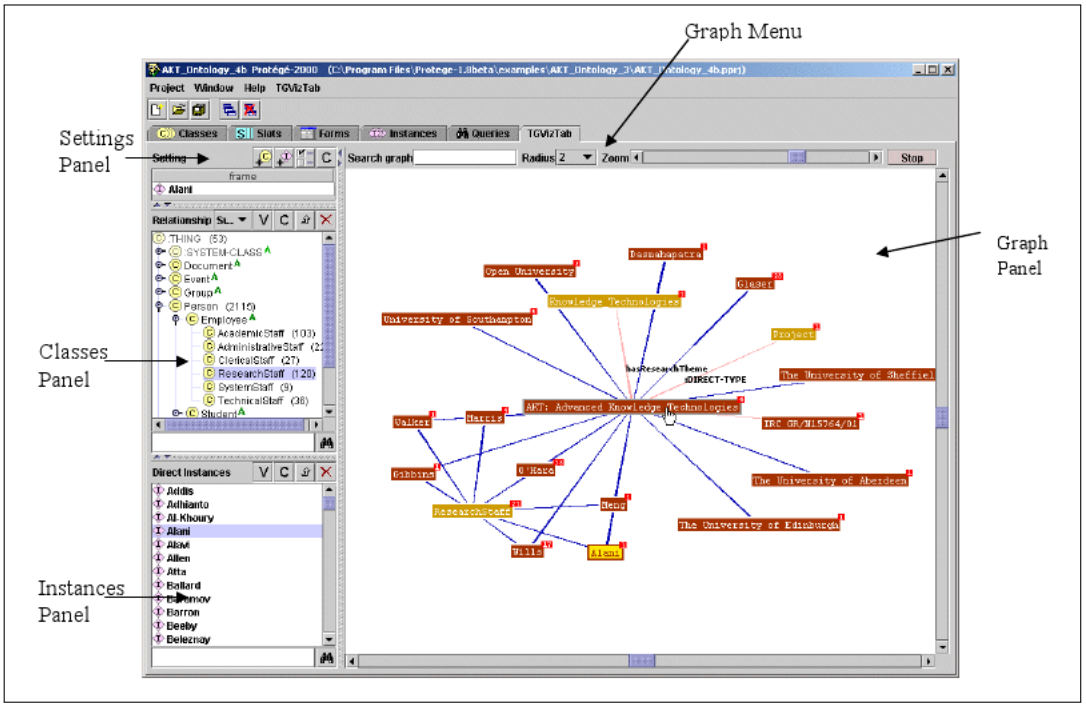


Figure 3.6: TGVizTab (taken from [11])

OntoSphere (Figure 3.7) is a three-dimensional ontology visualisation tool and the third entry in Table 3.1. Considering all the elements that form part of ontologies, Bosca *et al.* [19] argue that it can often be too restrictive to display in only two dimensions. Ontosphere uses three views to display the elements in an ontology to the user. Each of these distinct views employ their own visualisation methods and visualisation properties.

The *RootFocus* view presents a large sphere that displays a collection of concepts that are represented as smaller spheres. Role relations between the concepts in the view are shown, but not taxonomic information. An hyperbolic tree visualisation technique is used to display the smaller spheres within the big sphere. The RootFocus view uses colour and size as visualisation properties. Atomic classes (classes without any subclass) are smaller and have a distinct colour. Other classes

are highlighted in white with their size proportional to the cardinality of their subclasses.

The *TreeFocus* view visualises a selected concept within the ontology. The visualisation shows the selected concept within a taxonomic context. It also shows the selected concept's direct relations with other concepts in the ontology. A tree-like layout is used as a visualisation method. Colour is employed as a visualisation property to distinguish between the taxonomic and role relations of the selected concept.

The *ConceptFocus* view displays all the available information about a selected class. The selected concept's parents, ancestors, children and semantic relations can be found in this visualisation.

Even though OntoSphere does not focus on the visualisation of individuals in an ontology, it makes extensive use of visualisation properties and has various views to display the elements in an ontology. Therefore, we classify it as a more complex visualisation than TGVizTab.

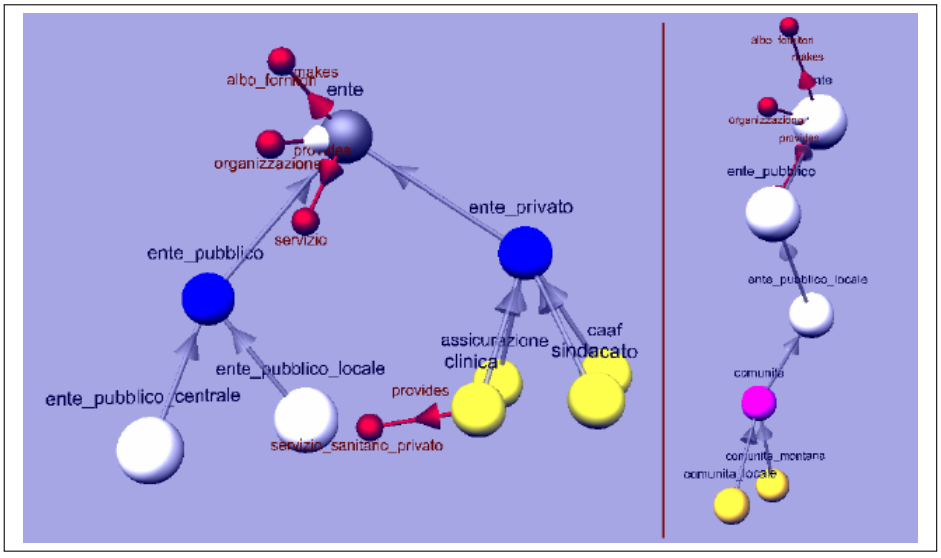


Figure 3.7: OntoSphere (taken from [19])

Like Protégé, Swoop (the fourth entry in Table 3.1) is an ontology editing tool. A two-dimensional visualisation method employed within Swoop is *CropCircles* [41]. While other ontology visualisation tools use incremental browsing and sub-graphs to visualise segments of the ontology to the user, CropCircles has the advantage of visualising the ontology as a whole. Consequently, it aids the user in retaining a context when inspecting individual elements in the ontology. CropCircles represents each class in the taxonomy as a circle. Children of a selected class is displayed as smaller circles within the parent circle. Children who are leaf nodes in the taxonomy

are displayed as a smaller sized circle than children who themselves have children. Child circles are placed in the parent circle in a spiral layout. Size and colour are used as visualisation properties in Swoop. A concept is represented as a circle and the size of this circles varies according to the size of the sub-tree in the taxonomy of this concept. All the circles in the initial CropCircle formation are displayed in grey. The user can select one of the classes with a mouse click. This selected circle and all the circles contained in it then take on white as colour. A search and selection pane is part of the CropCirle visualisation. Once a class is selected, this class and its children are displayed in the selection pane. The number of individuals of this class is displayed between brackets next to it in the selection pane. When one of the children classes is selected from the selection pane, the corresponding circle in the visualisation is highlighted in yellow. Figure 3.8 shows an example of how CropCircles are employed in Swoop. CropCircles does a very comprehensive

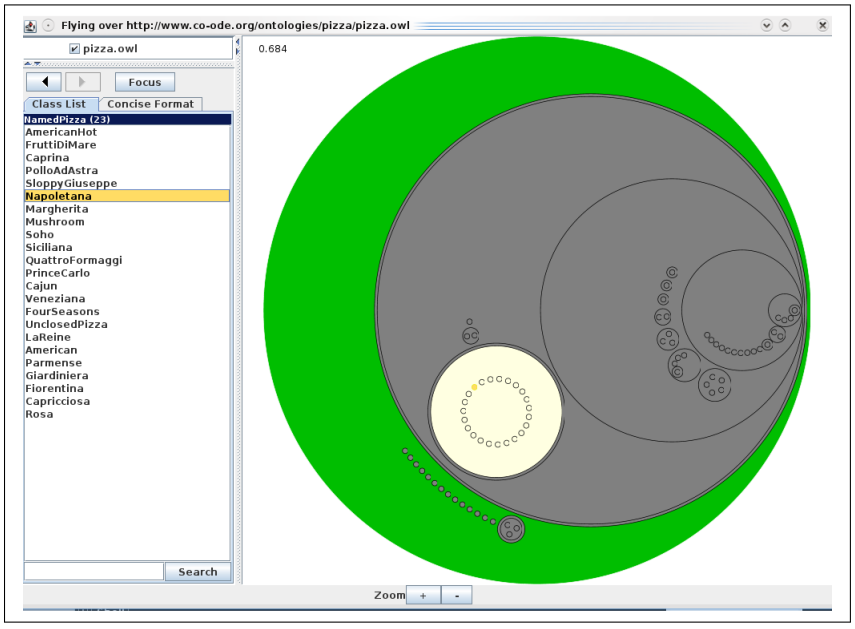


Figure 3.8: CropCircles in Swoop (Swoop screen-shot)

visualisation of an ontology and provides a view of an ontology in its entirety. Furthermore, it employs an effective visualisation method and makes extensive use of visualisation properties to portray elements in an ontology. OntoSphere, on the other hand, does visualisations in 3D and has several views. OntoSphere and CropCircles are similar in their levels of complexity, but we regard CropCircles as a more complex visualisation because of its innovative way of displaying the ontology in its entirety.

ClusterMap [23, 24] is a type of visualisation that has been employed in several software tools (the fifth entry in Table 3.1). ClusterMap visualises parts of the taxonomy of an ontology together with the individuals in the ontology. Fluit *et al.* [23] argue that there are few software tools that focus on instance-level visualisations, and of those none show the overlap (individuals belonging to more than one class). ClusterMap attempts to address this issue by visualising the individuals that belong to a certain class in the ontology and how these individuals relate to individuals of its child classes in the taxonomy of the ontology. This visualisation also shows which individuals belong to more than one class. Like TGVizTab, ClusterMap makes use of a variant of the spring-layout technique as a visualisation method. The result is that classes that share instances are located close to each other. Therefore, instances with the same or similar class memberships are also close to each other. Colour is used to differentiate individuals from different classes. Opacity is employed to show which individuals belong to more than one class. Individuals belonging to more than one class are displayed in a higher opacity. In summary, ClusterMap is a two dimensional software tool that focuses on the visualisation of individuals in an ontology. Figure 3.9 shows a ClusterMap example. ClusterMap is classified as the most complex of the visualisation tools

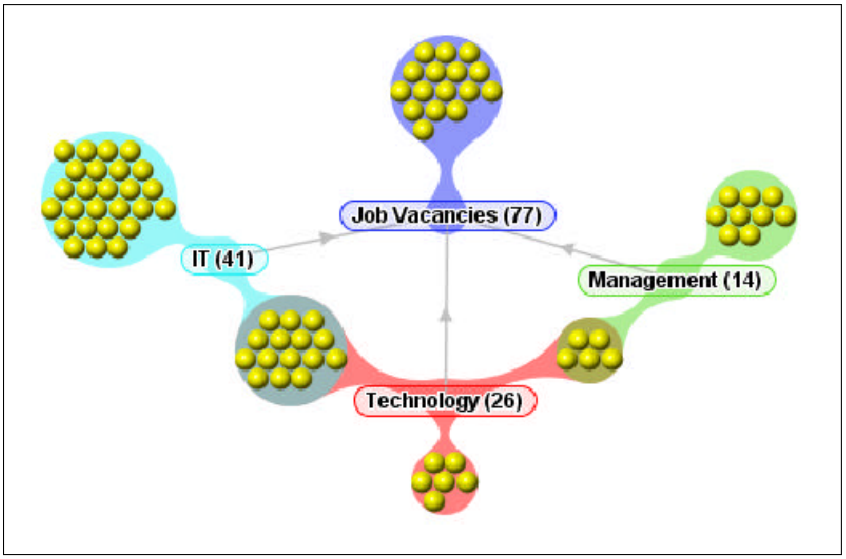


Figure 3.9: ClusterMap example (taken from [24])

discussed so far, because it applies a clustering technique on the data in an ontology. This technique provides us with additional information about the elements in the ontology.

VantagePoint [39] is a software tool that interactively visualises models of networked-home-environment ontologies (the seventh entry in Table 3.1). Van-

tagePoint is quite different from the other tools that we have studied so far, in the sense that it is not a general ontology visualisation tool. It visualises models of a specific ontology, namely networked home environments. It is important to note that the ontology itself is not visualised, but rather models of the networked home environment ontology. Most ontology visualisation tools do their visualisations in a graph-like manner with nodes and edges. However, VantagePoint generates a 3D visualisation of its models that is an accurate depiction of the real world. The visualisation method differs per model, as the model itself dictates where components should be placed on the screen. Visualisation properties do not play a pivotal role because the components and their colours also differ per model. VantagePoint also provides graphical querying functionalities. Visualisations in VantagePoint do help us to better understand the networked home environment ontology by visually displaying a variety of models of the ontology. Figure 3.10 depicts VantagePoint. VantagePoint is classified as the second most complex visualisation, because it

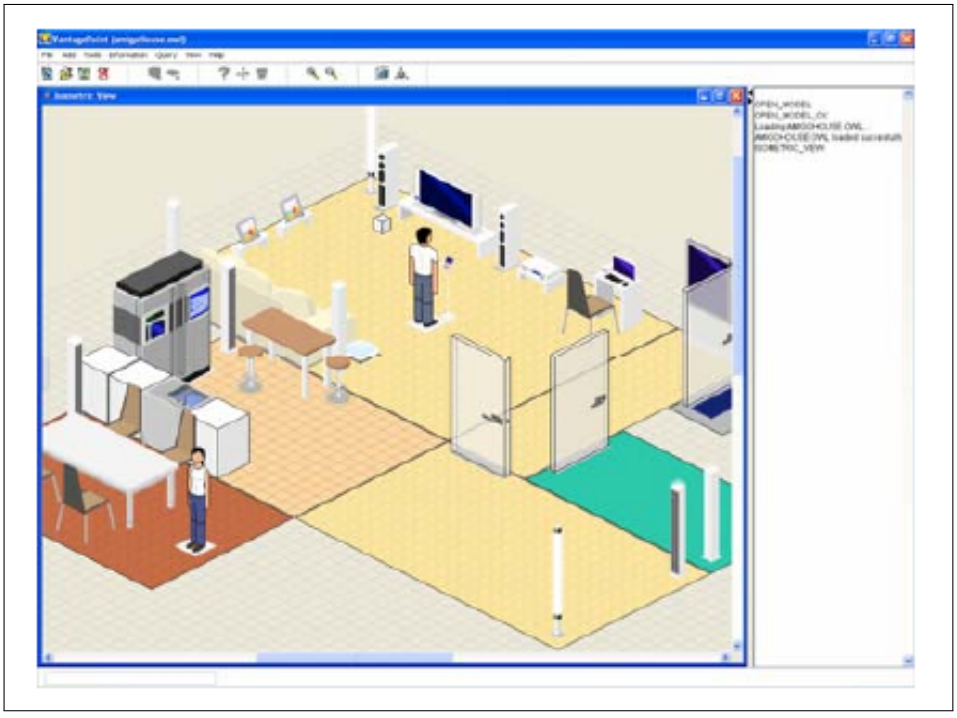


Figure 3.10: VantagePoint example (taken from [39])

touches on the realm of understanding. Even though it only operates on a single ontology, it creates fixed models of the ontology so that users can understand the networked home environment.

A summary of the visualisation methods and properties of the software tools that were evaluated can be found in the Table 3.2. When considering the *visu-*

Software tool	Visualisation method	Visualisation properties
OWLViz	tree layout	colour, shape, 2-dimensional
TGVizTab	spring layout	colour, 2-dimensional
OntoSphere	hyperbolic tree layout	colour, size, 3-dimensional
Swoop	cropcircles	colour, size, 2-dimensional
ClusterMap	spring layout	colour, opacity, 2-dimensional
VantagePoint	model specific	model-specific, 3-dimensional

Table 3.2: Summary of ontology visualisation tools

alisation methods and *visualisation properties* employed by the software tools in Table 3.2, it is not possible to distinguish which tools use more trivial visualisations and which tools use more complex visualisations. For example, *TGVizTab* and *ClusterMap* use the same visualisation method, but the former is the second most trivial visualisation while the latter is the second most complex visualisation. The key to distinguishing between trivial and complex visualisations lies in the underlying philosophy of the visualisation. By this we mean the way in which the information in an ontology is presented and adorned.

3.4 Understanding ontologies and model exploration

When using visualisation as an aid in the understanding of ontologies, two fundamental questions arise:

- *What* do we want to visualise in the understanding of ontologies?
- *How* do we want to visualise the understanding of ontologies?

We argue that the way in which we visualise the *understanding of ontologies* does not differ much from the way we visualise ontologies. We still use visualisation methods and visualisation properties to visualise the understanding of ontologies. Evaluation criteria will also measure the success of the implementation. However, *what* we want to visualise in the *understanding of ontologies*, changes fundamentally from *what* we want to visualise in ontologies (illustrated in Figure 3.11). Here, we do not merely visualise the ontological vocabulary, but we visualise techniques that aid the user in understanding ontologies. These techniques make use of the ontological vocabulary within the visualisation. In other words, the ontological vocabulary is employed within the context of an *ontology understanding technique*.

SuperModel [16, 17] is a visualisation tool that aids users in understanding ontologies. SuperModel was implemented as a software plug-in for Protégé. The technique that SuperModel employs is *model exploration*. SuperModel has much in

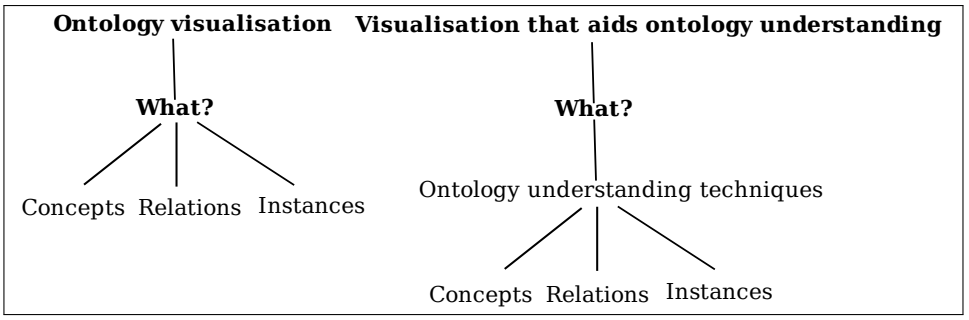


Figure 3.11: Ontology visualisation and understanding

common with VantagePoint. VantagePoint visualises models of a specific ontology, while SuperModel generates and visualises models of any ontology from a root concept. The user can also manipulate the model and test its consequent satisfiability. The idea of model exploration arose from the fact that a model that was generated could be changed within the bounds of the model space. The effect of these changes on the model can be *explored*. Models of an ontology can grow exceedingly large and this has an effect on the way SuperModel visualises models to users. Initially, SuperModel only visualises a part of a model to the user. An instance of the root concept is chosen by the user, and the direct relationships emanating from this instance are shown to the user. The user can expand the model by selecting elements in the visualisation. The part of the model that is visualised to the user is called the *model excerpt*. Figure 3.12 shows an example of such a model excerpt in SuperModel. As a visualisation method, SuperModel uses an expandable graph-like layout. Nodes in the graph that are expandable are highlighted in bold. Edges in the graph can be uni-directional or bi-directional.

In their comprehensive survey of existing ontology visualisation tools, Katefori *et al.* [37] remark that the representation of reasoning (or the effect of a reasoner on an ontology) in ontology visualisation is not satisfactory. OWLViz addresses the issue of reasoning representation to a limited extent, by visualising the inferred taxonomy. However, the interaction of SuperModel visualisations with the reasoner is much more extensive. We believe that SuperModel addresses the concern of Katefori *et al.* [37] for the following reasons:

- Models generated by SuperModel must be verified by the reasoner.
- Changes made to the model by the user have to be checked by the reasoner for satisfiability.
- Both of the above-mentioned are represented visually.

Furthermore, SuperModel's interaction with the reasoner and the visualisation of these interactions contribute in aiding the user to better understand the underlying ontology [16, 17].

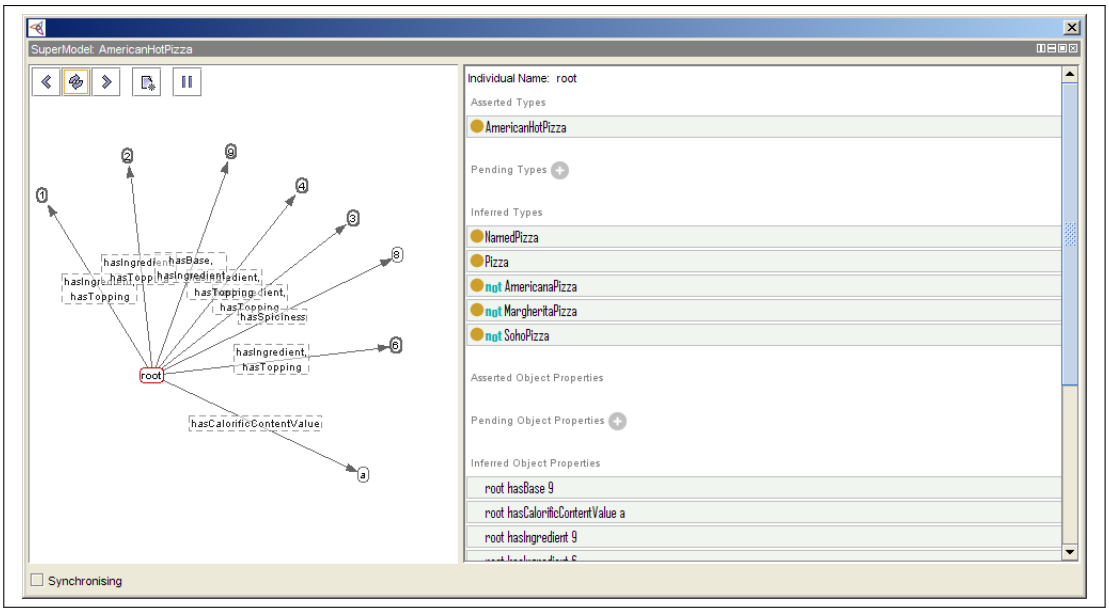


Figure 3.12: SuperModel plug-in (Protégé screen-shot)

3.5 Conclusion

This chapter illustrated how visualisation is beneficial in the process of understanding ontologies. The work of Bauer [16, 17] indicates that the usage of an ontology understanding technique in a visualisation is particularly helpful for enhancing ontology understanding.

We want to expand on the idea of SuperModel as a tool that aids the understanding of ontologies. In Chapter 5 we will define the notion of *ontology comprehension*. SuperModel is a software tool that will be classified in the context of ontology comprehension.

Four

Ontology Reasoning

4.1 Introduction

Ontology reasoning is important within the context of ontology comprehension. In order to thoroughly understand the ontology, it is instructive to understand the effect that the reasoner has on the ontology. Knowledge in the ontology, that is not immediately apparent, becomes evident when the reasoner is executed, because the consequences of assertions are made explicit. Additionally, ontology reasoners test the consistency of an ontology.

In this chapter, a background on ontology reasoning is given in Section 4.2. Here, the reasoning tasks within ontologies are discussed. Section 4.3 considers current literature that focuses on comprehending the effect of the ontology reasoner. Section 4.4 explains how software tools display the effect of the ontology reasoner. Section 4.5 concludes.

4.2 Background

Ontology reasoning is concerned with validating the axioms in an ontology and deducing new information from these axioms. Baader *et al.* [14] state that reasoning ensures the quality of an ontology, but it also exploits the rich structure of an ontology. Reasoning takes place at different stages of the ontology development life cycle.

The two main reasoning processes in ontologies are *validation* and *deduction* [46]. *Validation* is concerned with verifying the correctness of the axioms in an ontology from a mathematical perspective. This means that the axioms should not logically contradict each other. Validation often takes place in the design phase of the ontology.

In ontologies, *deduction* is the process of deriving new facts (axioms) from existing facts. Deduction can take place during the design of the ontology or after the ontology has been deployed. Table 4.1 illustrates how deduction works in an

OWL DL ontology. This example, a subset of the axioms of the Pizza ontology [3], indicates how deduction can work in an ontology. The axioms highlighted in blue are those that were responsible for the deduction, and the axioms highlighted in green are the newly deduced axioms. Here, the individual i_1 was asserted to be a `MargheritaPizza`. However, the ontology reasoner deduced that i_1 is also of type `CheesyPizza`, because `CheesyPizza` is the parent class of `MargheritaPizza`.

TBox	**ABox**
$Food \sqsubseteq \top$	$MargheritaPizza(i_1)$
$Pizza \sqsubseteq Food$	$PizzaBase(i_2)$
$PizzaBase \sqsubseteq Food$	$CheeseTopping(i_3)$
$PizzaTopping \sqsubseteq Food$	$TomatoTopping(i_4)$
	$hasBase(i_1, i_2)$
$Pizza \sqcap PizzaBase \equiv \bot$	$hasTopping(i_1, i_3)$
$PizzaBase \sqcap PizzaTopping \equiv \bot$	$hasTopping(i_1, i_4)$
$Pizza \sqcap PizzaTopping \equiv \bot$	
	Deduction:
$MargheritaPizza \sqsubseteq Pizza$	$\therefore MargheritaPizza \sqsubseteq CheesyPizza$
$CheeseTopping \sqsubseteq PizzaTopping$	$\therefore CheesyPizza(i_1)$
$TomatoTopping \sqsubseteq PizzaTopping$	
$CheeseTopping \sqcap TomatoTopping \equiv \bot$	
$MargheritaPizza \sqsubseteq \exists hasTopping.CheeseTopping$	
$MargheritaPizza \sqsubseteq \exists hasTopping.TomatoTopping$	
$MargheritaPizza \sqsubseteq$ $\forall hasTopping.(TomatoTopping \sqcup CheeseTopping)$	
$CheesyPizza \sqsubseteq Pizza$	
$CheesyPizza \equiv \exists hasTopping.CheeseTopping$	

Table 4.1: Example of deduction

Reasoning in ontologies will now be discussed from the perspective of DLs. The most important reasoning tasks (*knowledge base satisfiability*, *concept satisfiability*, *instance checking* and *subsumption* [46]) will be considered in DLs within the context of validation and deduction.

Knowledge base satisfiability is a validation task which ensures that axioms in a knowledge base can be interpreted in such a way that none of them are violated [46]. When there is such an interpretation of the knowledge base, we have a model of the knowledge base. Inconsistent knowledge bases without a model are not useful for deduction.

Concept satisfiability is a validation task that verifies whether *concepts* can have *instances* [46]. This in essence is a check to see whether an abstract concept can have a concrete implementation.

Instance checking is both a validation and deduction task. As a validation task, it checks to see whether an instance belongs to a given (defined) concept in the knowledge base. As a deduction task, it can be used to infer to which type (concept) an instance belongs.

Subsumption is a deduction task and it checks whether a given concept is a sub-concept of another one [46].

4.3 Understanding the workings of the ontology reasoner

Wang *et al.* [47] studied the workings of the ontology reasoner in the context of debugging (repairing broken ontologies). They describe it as a non-trivial task for two reasons. Firstly, the axioms (in an ontology) can have wide ranging effects that are hard to predict. Secondly, unsatisfiability propagates. This means that a single root error can cause many classes to be marked as unsatisfiable.

They propose a heuristic approach to repair broken ontologies. This approach is reasoner-independent and starts with an unsatisfiable class in an ontology. The consequent processes determine the set of axioms in the ontology that have resulted in the incoherency. The final part of the approach analyses the conflicting axioms in order to determine the root cause of the incoherency. We consider two other studies that expand on these ideas.

Parsia *et al.* [40] discuss an approach for debugging OWL ontologies within the context of the Swoop ontology editor. Like Wang *et al.*, their discussion focuses on the diagnosis and correction of unsatisfiable concepts. They argue that the detection of an unsatisfiable concept is easy, but the challenge lies in determining *why* it is so. They follow an approach of diagnosing unsatisfiability and exploring remedies.

Kalyanpur *et al.* [35] discuss enhancements to the ontology debugging techniques that have been considered so far. They propose an approach whereby they generate repair solutions based on strategies that were used to rank erroneous axioms. Their strategy is quite specific in the sense that it detects the *part* of an axiom that is erroneous. They suggest a rather elaborate repair solution that is based on a modification to Reiter's algorithm. Interested readers are referred to Kalyanpur *et al.* [35] for the details.

The above mentioned literature focus on understanding why a given ontology is incoherent, so that it can be repaired. However, it can also happen that the given ontology is coherent and that the users of such an ontology simply want to understand the ontology and the effect that the reasoner has on the ontology.

Kalyanpur *et al.* [34] consider this scenario. They use similar techniques to that of the ontology debugging process, but the focus is on explaining why the ontology reasoner made certain entailments (inferences) and not on repairing a broken ontology.

4.4 Ontology reasoning and software tools

Certain ontology software tools discussed in Chapter 3 show the effect of entailments. For example, OWLViz [2, 37] visualises the inferred class hierarchy of an ontology. The discussions in Section 4.3 also relates to the effect of entailments, but goes much further. Here, it is not only about showing the effect of entailments, but it is about showing *how* and *why* the ontology reasoner made certain inferences.

4.5 Conclusion

The most important reasoning processes in ontologies are validation and deduction. Validation verifies the correctness of the ontology while deduction derives new facts (axioms) from existing facts.

Further discussions in this chapter illustrated that much of the work relating to understanding the effect of entailments originated from difficulties relating to the repair of incoherent ontologies. Similar techniques are used to understand entailments made by the ontology reasoner in coherent ontologies.

Ontology debugging and explanation do not only show the effect of entailments, but show *how* and *why* the ontology reasoner made certain inferences.

Chapter 2, Chapter 3 and this chapter serve as a theoretical framework. The next chapter, Chapter 5, describes an ontology comprehension framework that aims to answer the first sub research question.

Five

Ontology comprehension

5.1 Introduction

This chapter is based on the literature survey that was done in Chapter 3. Within this chapter we develop an ontology comprehension framework by categorising concepts from the literature survey conducted in Chapter 3.

In Section 5.2 we formalise an ontology comprehension framework, and conclude with a discussion of model exploration within the context of this framework.

We continue, by defining *concept relationship analysis* (CRA) as our proposed ontology understanding technique and Section 5.3 explains what CRA is. Section 5.4 contains a comparison between model exploration and CRA. Section 5.5 explains the relationship between ontology comprehension and visualisation.

This chapter addresses our first sub research question (Q_1), namely, *How can we construct an ontology comprehension framework wherein we can classify the approaches related to ontology comprehension?*

5.2 Ontology comprehension

The term *ontology comprehension* is used in different contexts with different meanings by different authors (for example [21, 25, 38]). For this reason it is important to clarify the meaning of ontology comprehension within the context of this work. We find the definition of Gibson *et al* [25] to be closely related to our work:

> *We outline* `ontology comprehension` *as the interaction between human agents and the knowledge expressed in an ontology.*

In our context, we define ontology comprehension as follows:

Definition 5.1 *Ontology comprehension is a collection of techniques that facilitate the understanding of ontologies.* □

An ontology understanding technique is an abstract idea that can have many implementations. For example, two computer programs can implement the same technique in different ways.

Ontology understanding techniques can operate independently, or can interact to enhance understanding. A depiction of our ontology comprehension framework is given in Figure 5.1.

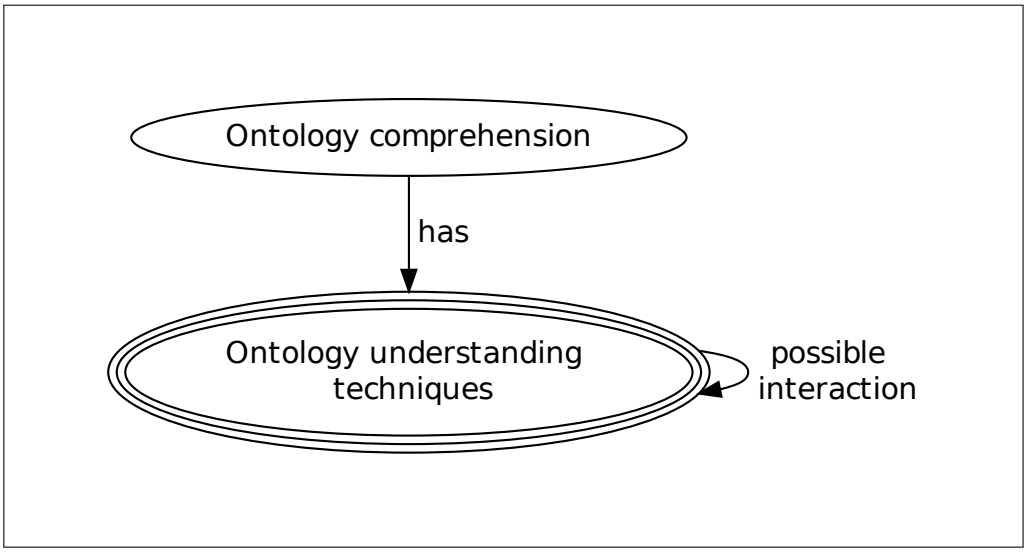


Figure 5.1: Ontology comprehension framework

Within the framework for ontology comprehension, we now consider model exploration. In Chapter 3 we identified model exploration as an ontology understanding technique. SuperModel is a specific implementation of model exploration. We can also say that SuperModel is one instantiation of model exploration (there can be other instantiations of model exploration). Figure 5.2 illustrates this situation.

Definition 5.2 *An ontology understanding technique is an abstract idea with the aim of facilitating the process of understanding ontologies.* □

5.3 Concept relationship analysis (CRA)

In this section we discuss two important questions:

- What is CRA?
- What valid grounds are there to believe that CRA will contribute to ontology comprehension?

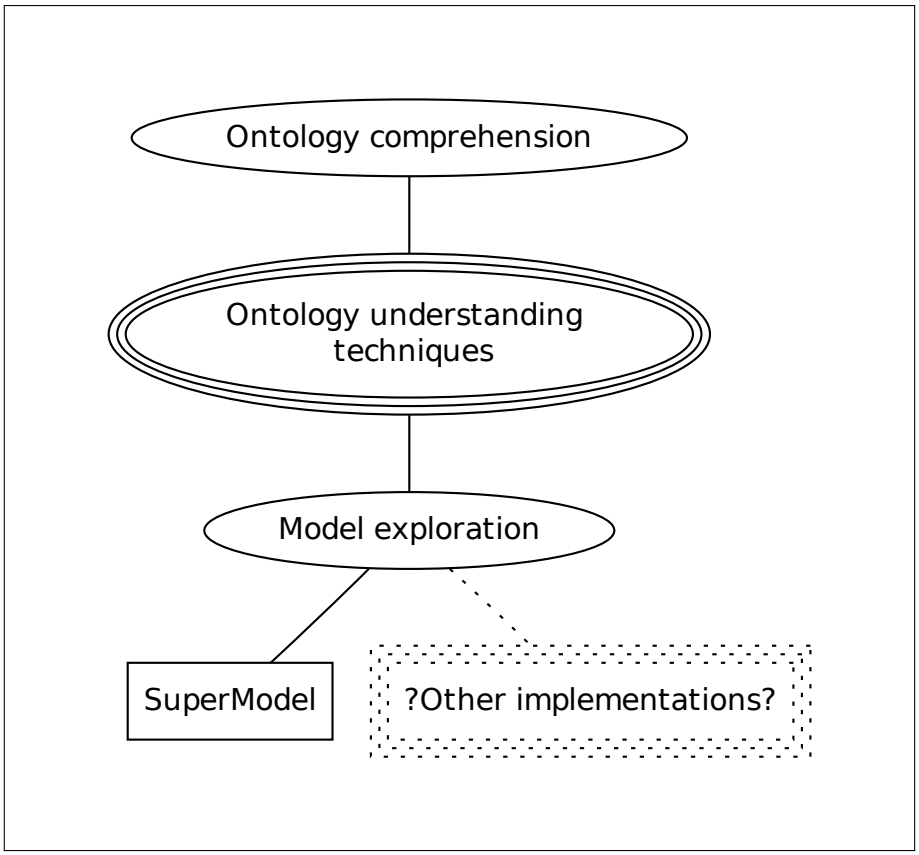


Figure 5.2: SuperModel in the ontology comprehension framework

Firstly, we define CRA as follows:

Definition 5.3 *Concept relationship analysis (CRA) is an ontology understanding technique that aims to facilitate the process of understanding ontologies by analysing the relationships amongst concepts in an ontology.* □

The importance of understanding the relationships that entities have with each other has been prominent in the database world for decades. For example, relational databases is a widely used database paradigm. In these databases, data is modeled and stored to reflect the relationships between entities. Further support for our argument can be found in the example of the Venn diagram. Venn diagrams are widely used to understand the relationship different sets have with one another. An example of a Venn diagram can be found in Figure 5.3.

In ontologies, we can make entailments about the ontology that provide us with additional information on how concepts relate to each other. The analysis of concept relationships should take the effect of these entailments into account.

As an ontology understanding technique, CRA is an abstract idea. An abstract idea can have many possible implementations. Figure 5.4 illustrates the place of

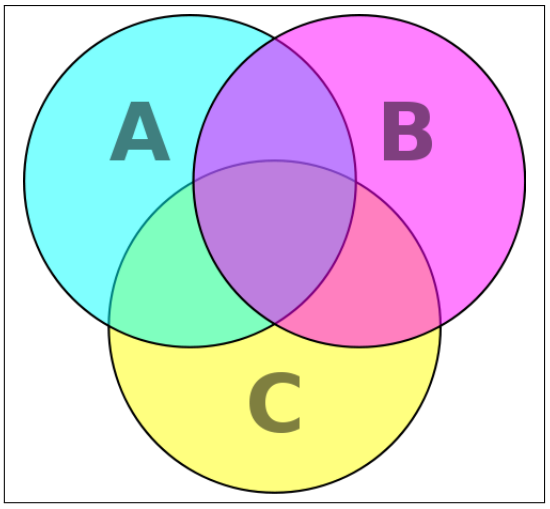


Figure 5.3: Example of a Venn diagram (taken from [10])

CRA in the ontology comprehension framework.

5.4 Model exploration and CRA

Model exploration is a known ontology understanding technique. CRA is our proposed ontology understanding technique. It differs from model exploration, but also aims to facilitate the understanding of ontologies.

We highlight the most prominent differences between these two techniques. Model exploration works on the principle of iteratively expanding a model of the ontology from a chosen root concept [16, 17]. If the aim is to find a goal concept, it may take many iterations to find it, or it may not be found at all. CRA, on the other hand, is more goal driven in the sense that a root and a goal concept are stated upfront. Then we analyse the way the root and the goal concept relate to each other. CRA observes two concepts in the ontology and analyses their relationship, while model exploration generates a model of the ontology from a single concept, called the root concept. All the elements within the model that is being explored, are individuals (instances of concepts). Consider the following toy ontology together with Figure 5.5 as an example that illustrates model exploration and CRA:

$$C \sqsubset \exists hasRelationship.D$$

$$D \sqsubset \exists hasRelationship.E$$

$$E \sqsubset \exists hasRelationship.F$$

$$C \sqsubset \exists hasRelationship.G \ .$$

Figure 5.5 is divided into two parts. The first part of the image depicts the above-

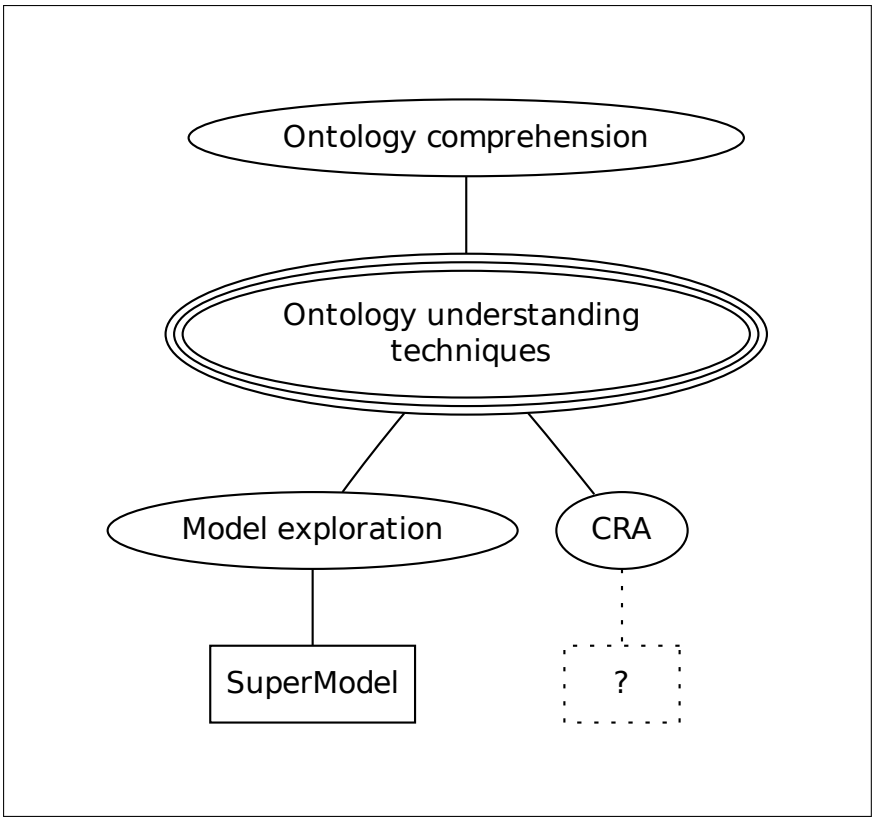


Figure 5.4: CRA in the framework

mentioned toy example from the point of view of model exploration. Here, the elements in the toy example are linked to each other as instances (individuals). The second part of the image depicts the same toy example from the point of view of CRA. Here, the elements in the toy example are linked to each other as concepts. The arrows in the image represent the *hasRelationship* role. □

Future research efforts could investigate the possibility of interaction between model exploration and CRA to enhance ontology comprehension.

5.5 Ontology comprehension and visualisation

In Chapter 3 we presented a detailed discussion on the importance of visualisation in ontology understanding. We used a selection of ontology visualisation tools as a starting point for ontology comprehension.

How is ontology comprehension related to visualisation? Visualisation is used as an aid when implementing an ontology understanding technique. Other aids, such as audio, can also be used during an implementation. Figure 5.6 depicts the role of visualisation in the ontology comprehension framework. Here, SuperModel

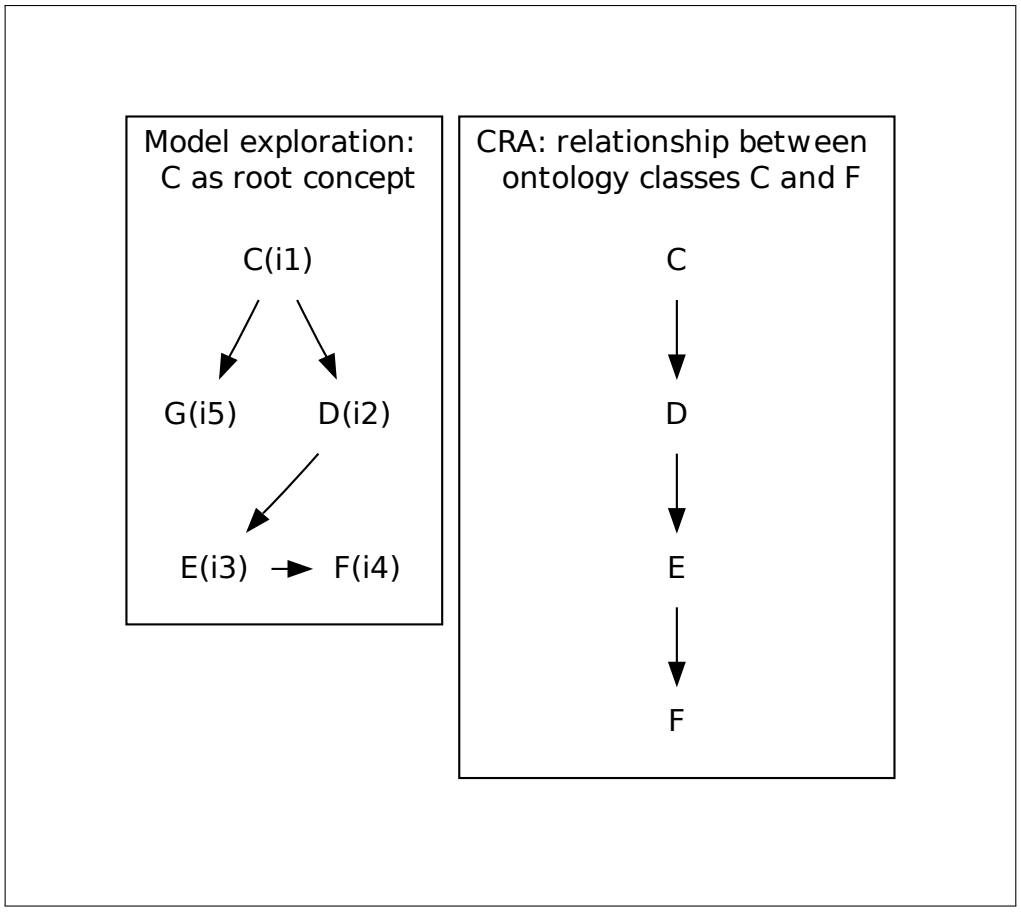


Figure 5.5: Model exploration and CRA

is shown as an implementation of model exploration (an ontology understanding technique). *Visualisation* is depicted as an implementation aid that is employed by SuperModel.

5.6 Conclusion

In this chapter, we presented our proposed ontology comprehension framework. Within this framework we are able to classify software and techniques that facilitate the understanding of ontologies. In addition, we developed a new ontology understanding technique within this framework, namely, Concept Relationships Analysis (CRA). CRA aims to facilitate the understanding of ontologies by analysing concept relationships. Finally, we illustrated how visualisation relates to ontology comprehension. Here, we argued that visualisation can be seen as an aid in the implementation of an ontology understanding technique.

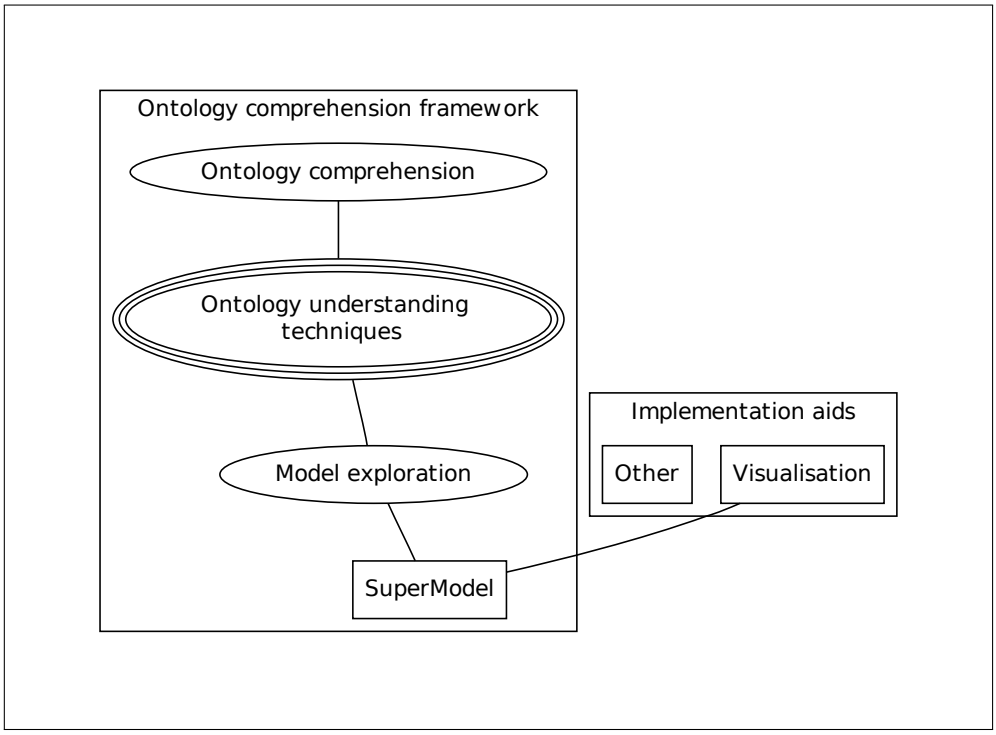


Figure 5.6: Role of visualisation in the framework

Six

PathViz

6.1 Introduction

In Chapter 5 we developed an ontology understanding technique, called concept relationship analysis (CRA). CRA, like model exploration, is an abstract idea that can have many concrete implementations. To test the viability of CRA, we develop and evaluate a concrete implementation. This implementation is called PathViz and Figure 6.1 illustrates it within the ontology comprehension framework as defined in Chapter 5.

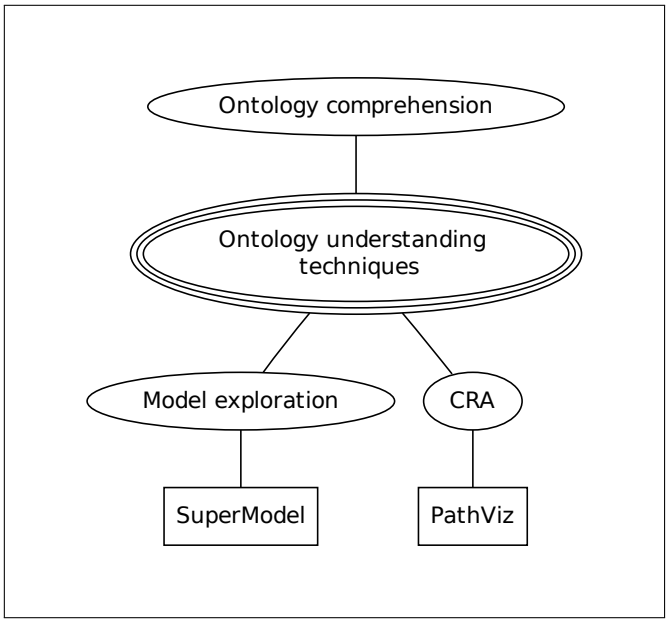


Figure 6.1: PathViz in the ontology comprehension framework

Section 6.2 explains the PathViz implementation and Section 6.3 gives a formal description of the process. Section 6.4 explains how the paths in PathViz can be

interpreted. The assumptions that were made in the implementation can be found in Section 6.5. We show how PathViz is related to the notion of models, before we conclude in Section 6.7.

This chapter addresses our second sub research question (Q_2), namely, *How can we apply path visualisation techniques to comprehend subsumption and existential relationships between concepts in an ontology?*

6.2 PathViz

We propose PathViz as an implementation of CRA and we want to illustrate that PathViz complies with the requirements of CRA. Previously, we defined CRA to be the analysis of the relationships amongst concepts in an ontology. The discussion of PathViz in this section should clarify the compliance of PathViz with the definition of CRA.

In Chapter 2 and Appendix A, Protégé is described as a software tool for building and using ontologies. Protégé was designed so that it is easy to extend the functionality with *plug-ins*. Appendix A elaborates on how plug-ins operate in Protégé.

PathViz is a Protégé plug-in that uses a step-based wizard approach to aid users in understanding ontologies (technical details in Appendix B). Graphs of both the asserted and inferred taxonomies are created, subsequently referred to as the asserted graph and the inferred graph. These graphs are constructed from two types of relationships in the ontology as described in Chapter 2, namely, subsumption and existential relationships. Firstly, we explore the subsumption relationship. This part of the graph construction takes the taxonomy of the ontology and places it in a graph structure (concept names are linked to each other with *is-a* relationships). Secondly, we explore existential relationships in the ontology. Existential relationships that form part of concept names in the ontology are also added to our graph structure. Note that this process is followed for both asserted and inferred taxonomies of the ontology. An example of an asserted and inferred graph can be seen in Figure 6.2. When using PathViz, the user selects two concept names in the ontology (Figure 6.3). PathViz computes a *set of paths* between the two selected concept names for both the asserted and inferred graphs. These paths are then rendered visually to the user. This is depicted in Figure 6.4. The user can iterate through both sets of paths (obtained from the asserted and inferred graphs) and visually compare them. Each rendered path has a status label to indicate whether that specific path can also be found in the path list from the other graph. For example, if we are looking at one of the paths in the set obtained from the inferred graph, we will be able to see if there is a similar path in the set of paths obtained from the asserted graph.

PathViz also intends to show what influence the reasoner has on the way con-

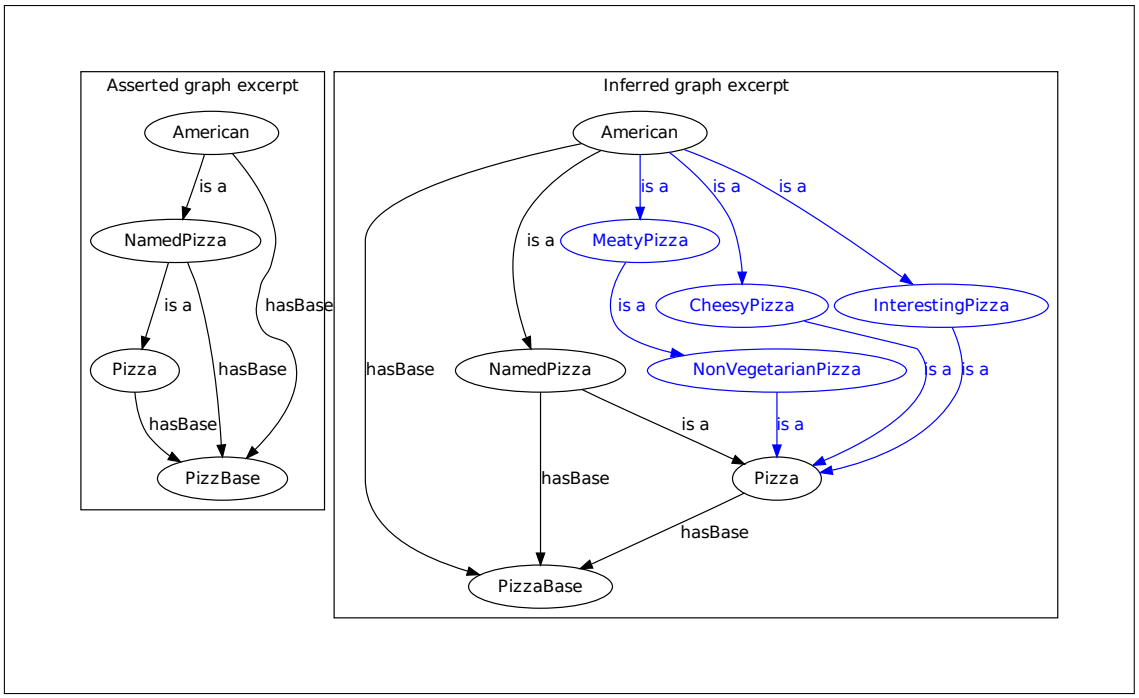


Figure 6.2: Asserted and inferred graphs (extracted from [3])

cept names in the ontology relate to each other. To measure this, we define two measurement instruments: *Path Cardinality Ratio* (PCR) and *Path Simplicity Ratio* (PSR). The next section explains the PathViz process formally before Chapter 7 elaborates on these two measurement instruments.

6.3 Formal representation of PathViz process

In this section we present the PathViz process formally. Consider an ontology O, where T is the $TBox$ and A is the $ABox$:

$$O =< T, A > .$$

Let C and R refer to a set of concept names and roles respectively in the ontology. Axioms in a $TBox$ are constructed by using C, R and boolean constructors.

After executing a tableaux reasoner over the ontology O, we obtain a new set of terminological axioms, T', via a function, B:

$$T' = B(O) .$$

It follows directly that T is a subset of T':

$$T \subseteq T' .$$

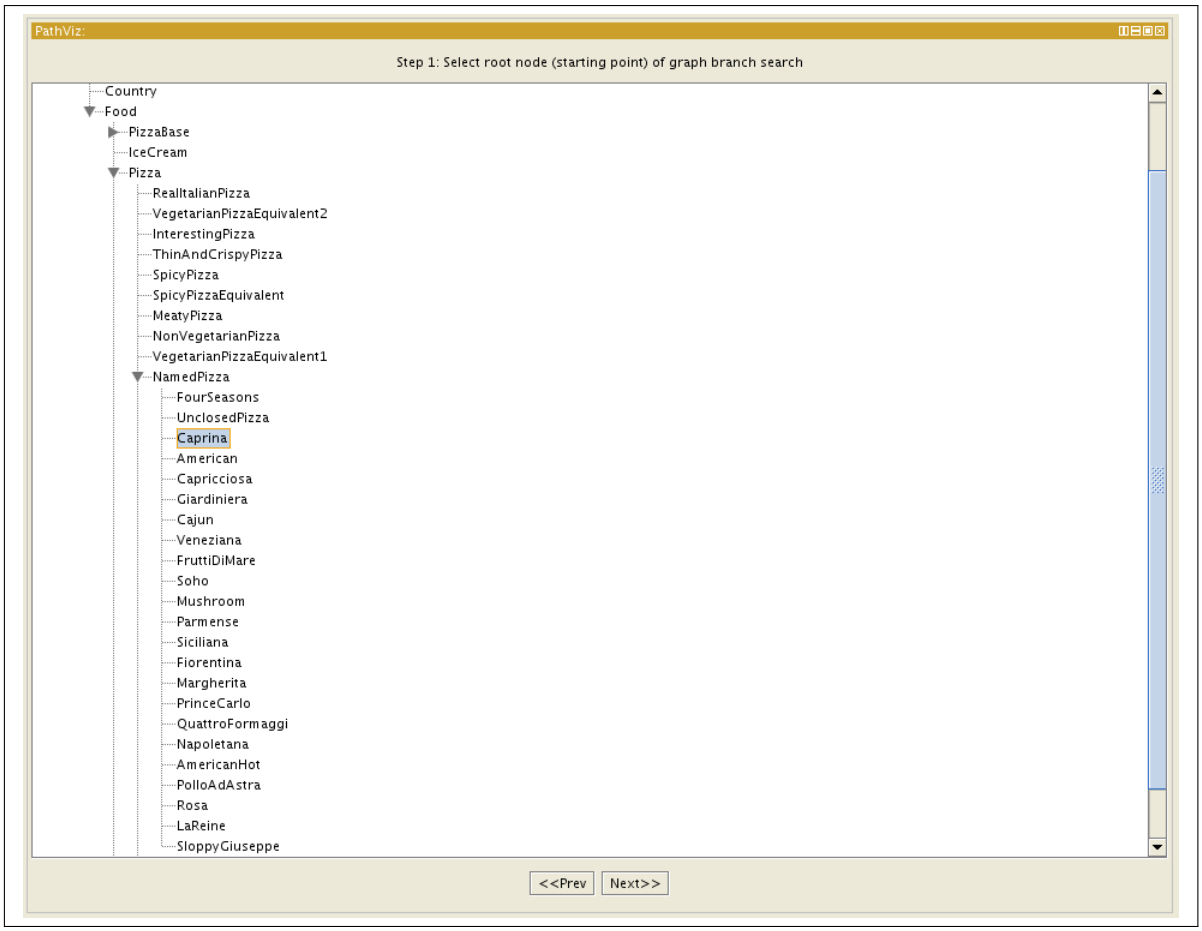


Figure 6.3: First step of *PathViz* (select root concept name)

A directed graph is an ordered pair

$$G := (V, E),$$

where V is a set of nodes (vertices) and E is a set of ordered pairs of nodes (vertices).

Consequently, we use T to construct the asserted graph, $G(V, E)$, where

$$V = C \text{ and}$$

$$E = Q(V) \ .$$

Here, Q is a function that returns a set of objects that links two concept names. This set is obtained by extracting the asserted taxonomy and the existential relationships therein, from the ontology.

Similarly, the inferred graph, $G'(V', E')$, is constructed from T' where

$$V' = C \text{ and}$$

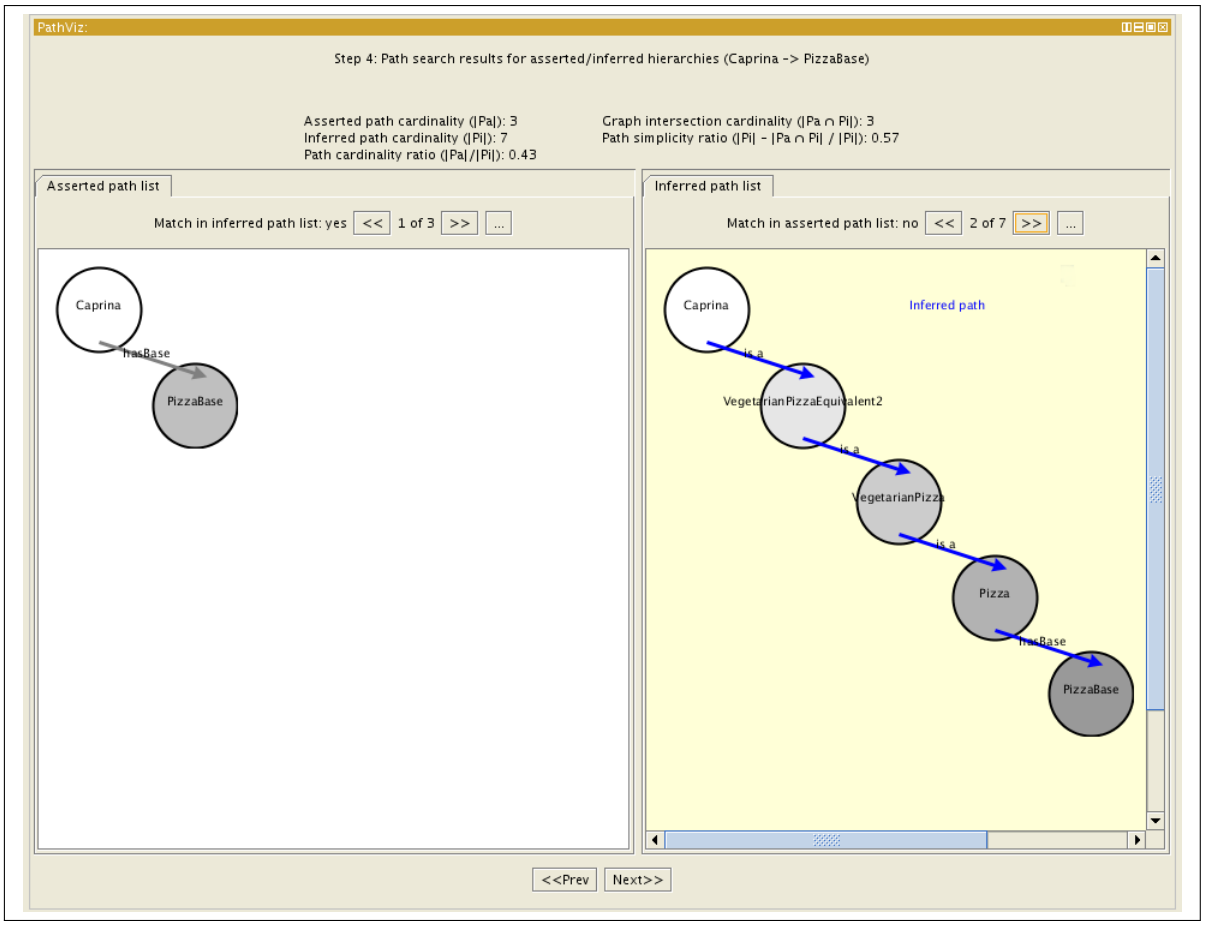


Figure 6.4: *PathViz* search results

$$E' = Q'(V') \ .$$

Suppose we now choose an arbitrary v_r as a root concept name and an arbitrary v_g as goal concept name from C. We define a function F to compute a set of paths P between v_r and v_g in graph G_i:

$$P := F(v_r, v_g, G_i) \ .$$

Every path in P is a list of edges:

$$\{p|p \in P\}, p := (x_1, r_1, y_1), (y_1, r_2, y_2), ..., (y_{n-2}, r_{n-1}, y_{n-1}), (y_{n-1}, r_n, y_n) \ .$$

Here, x and y (with $x \neq y$) refer to concept names in an ontology and r refers to the relationship that links these two concept names. The relationship, r, can either be a subsumption relationship or an existential relationship. As an illustration, we show the following example of a path from the well-known pizza ontology [3] where

$v_r = AmericanPizza$ and $v_g = PizzaBase$:

$$p = \{(AmericanPizza, \text{is a}, CheeseyPizza), (CheesyPizza, \text{is a}, Pizza), (Pizza, hasBase, PizzaBase)\} .$$

For a fixed v_r and v_g we proceed to compute a set of paths P_a and P_i for the asserted and inferred graphs respectively, where

$$P_a = F(v_r, v_g, G(V, E)) \text{ and}$$

$$P_i = F(v_r, v_g, G'(V', E')) .$$

We conclude by using the sets P_a and P_i for further analysis.

6.4 Interpretation of paths

Certain types of paths in PathViz can be expressed as a DL sentence. In this regard, we focus on two types of paths specifically. Firstly, the focus is on paths that have only subsumption relationships and secondly the focus is on paths that have only existential relationships.

Consider Figure 6.5. Here, we see that A_1 is subsumed by A_3 and A_2 is also subsumed by A_3.

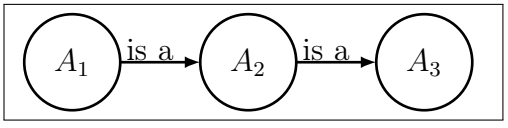


Figure 6.5: $A_1 \sqsubseteq A_3$

However, we can express this meaning in general, where a path has n nodes and all of the nodes are connected to each other with subsumption relationships (Figure 6.6). This can be expressed as

$$\{A_i \sqsubseteq A_j | 1 \leqslant i \leqslant n-1, i < j \leqslant n\} .$$

Let $a = \{A_i \sqsubseteq A_j | 1 \leqslant i \leqslant n-1, i < j \leqslant n\}$ and let Cn be the function that computes the consequences of an axiom. Then we observe that

$$\begin{aligned} Cn(a) = A_1 &\sqsubseteq A_2 \\ A_1 &\sqsubseteq A_3 \\ A_1 &\sqsubseteq A_4 \\ &\vdots \\ A_1 &\sqsubseteq A_{n-1} \\ A_1 &\sqsubseteq A_n; \\ &\vdots \\ A_2 &\sqsubseteq A_3 \\ A_2 &\sqsubseteq A_4 \\ A_2 &\sqsubseteq A_5 \\ &\vdots \\ A_2 &\sqsubseteq A_{n-1} \\ A_2 &\sqsubseteq A_n; \quad \text{and} \\ &\vdots \\ A_{n-1} &\sqsubseteq A_n \ . \end{aligned}$$

□

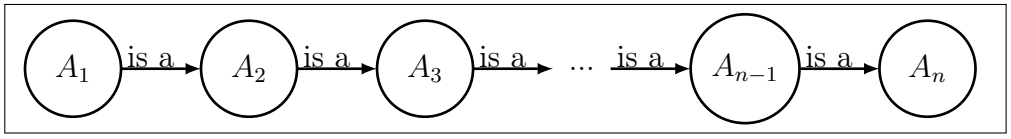


Figure 6.6: $\{A_i \sqsubseteq A_j | 1 \leqslant i \leqslant n-1, i < j \leqslant n\}$

Secondly, we consider paths that only have existential relationships. If a path has more than two nodes, then concept names in the paths can be linked to each other via role chaining. In Figure 6.7 the concept A_1 can be linked to the concept A_3 via role chaining and is then represented as $A_1 \sqsubseteq \exists(R_1 \circ R_1).A_3$.

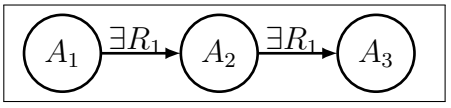


Figure 6.7: $A_1 \sqsubseteq \exists(R_1 \circ R_1).A_3$

We can express this generally, where a path has n nodes and all of them are linked to each other with the same existential role relationship (Figure 6.8). For simplicity sake, we denote $R_1 \circ R_1$ as R_1^2. Now we can express these paths with the same existential role relationship as

$$\{A_i \sqsubseteq \exists(R_1^{j-1}).A_j | 1 \leqslant i \leqslant n-2, i+2 \leqslant j \leqslant n\} \ .$$

Let $a = \{A_i \sqsubseteq \exists(R_1^{j-1}).A_j | 1 \leqslant i \leqslant n-2, i+2 \leqslant j \leqslant n\}$ and let Cn be the function

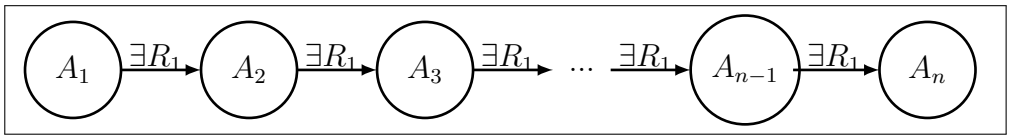


Figure 6.8: $\{A_i \sqsubseteq \exists(R_1^{j-1}).A_j | 1 \leqslant i \leqslant n-2, i+2 \leqslant j \leqslant n\}$

that computes the consequences of an axiom, then we observe that

$$\begin{aligned}
Cn(a) = A_1 &\sqsubseteq \exists(R_1^2).A_3 \\
A_1 &\sqsubseteq \exists(R_1^3).A_4 \\
A_1 &\sqsubseteq \exists(R_1^4).A_5 \\
&\vdots \\
A_1 &\sqsubseteq \exists(R_1^{n-2}).A_{n-1} \\
A_1 &\sqsubseteq \exists(R_1^{n-1}).A_n; \\
&\vdots \\
A_2 &\sqsubseteq \exists(R_1^3).A_4 \\
A_2 &\sqsubseteq \exists(R_1^4).A_5 \\
A_2 &\sqsubseteq \exists(R_1^5).A_6; \\
&\vdots \\
A_2 &\sqsubseteq \exists(R_1^{n-2}).A_{n-1} \\
A_2 &\sqsubseteq \exists(R_1^{n-1}).A_n; \quad \text{and} \\
&\vdots \\
A_{n-2} &\sqsubseteq \exists(R_1^{n-1}).A_n \ .
\end{aligned}$$

□

One caveat to look out for is paths with more than one type of existential role relationship. Such paths cannot be generalised into a role chaining DL sentence as was described above. Figure 6.9 gives an example of such a case. Here, the roles

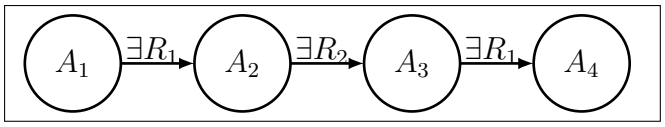


Figure 6.9: More than one type of existential role in a path

that connect the concept names in the path are not the same. The role R_1 connects A_1 and A_2, while the role R_2 connects A_2 and A_3.

Table 6.1 summarises how paths in PathViz can be interpreted. Here, (1) and (2) can be expressed as a DL sentence. This means that there are DL sentences to describe paths that only have subsumption relationships and paths that only have existential relationships of the same type.

Path description	**Consequence**
(1) Only subsumption relationships	$\{A_i \sqsubseteq A_j \vert 1 \leqslant i \leqslant n-1, i < j \leqslant n\}$
(2) Only existential relationships of the same type	$\{A_i \sqsubseteq \exists(R_1^{j-1}).A_j \vert 1 \leqslant i \leqslant n-2, i+2 \leqslant j \leqslant n\}$

Table 6.1: Summary of path semantics in PathViz

Entry (2) in Table 6.1 describes a DL sentence that uses role chaining. In Section 2.3.3 in Chapter 2 the different varieties of description languages were described. Some description languages, such as $\mathcal{SROIQ}$, support role chaining. Moreover, there are description languages for custom applications, such as GRAIL for GALEN, that support role chaining.

6.5 Implementation assumptions and restrictions

When we construct the asserted and inferred graphs, we consider the subsumption and existential relationships within the ontology. For the PathViz implementation we include relationships with minimum and exact cardinality, because they imply that there is an existential relationship. We illustrate this with two examples below:

$$(\geq 5 hasTopping) \sqcap \forall hasTopping.CheeseTopping \vDash \exists hasTopping.CheeseTopping$$

and

$$(= 1 hasTopping) \sqcap \forall hasTopping.MeatTopping \vDash \exists hasTopping.MeatTopping\ .$$

We did not consider universal restrictions ($\forall$) during the graph construction. Although $\forall r.C$ implies that all relationships r should be with a class of type C, it does not necessarily mean that such a relationship *has* to exist. In addition,

universal restrictions are often used in conjunction with existential relationships as closure axioms:

$$\exists hasTopping.CheeseTopping \sqcap \forall hasTopping.CheeseTopping \ .$$

A closure axiom is used to ensure that at least one relation of a specific type must exist and that all relations between two concepts are of this type.

Undeniably, universal restrictions play an important role in ontologies, and future investigations should focus on how it can best be represented in PathViz.

The presence of cycles in the asserted or inferred graph can pose problems. If these cycles are not detected the computation time of the asserted or inferred graph will be infinite. This problem is illustrated in the following example:

$$A \sqsubseteq \exists B$$
$$B \sqsubseteq \exists A$$

In order to circumvent the problems related to cycles in graphs, a cycle detection algorithm was implemented in PathViz in the graph construction phase.

6.6 PathViz and models

Although PathViz does not generate models of an ontology, a discussion on how it relates to models is appropriate, because it will give an idea of how PathViz relates to SuperModel (a software tool that does generate models). In Chapter 5 we argued that ontology understanding techniques can interact with each other to enhance comprehension. This discussion will serve as an example. It will argue that CRA (PathViz) and model exploration (SuperModel) can be used together.

In an ontology O, it is possible to devise a general algorithm to construct an ABox, A, from every path, p, in the sets of paths, P_a and P_i, such that

$$O \models A \ .$$

For example, consider the pizza ontology [3] and suppose that we have the following path p:

$$p = \{(Margherita, hasBase, PizzaBase)\} \ .$$

The first step is to construct an ABox from the information available in the path:

$$Margherita(i_1)$$
$$PizzaBase(i_2)$$
$$hasBase(i_1, i_2) \ .$$

In the second step, we go through the ontology to find other dependencies that we need to complete the ABox. In this example we find that a margherita pizza needs to have a mozzarella topping and a tomato topping. We complete the ABox by adding this:

$$
\begin{aligned}
&Margherita(i_1)\\
&PizzaBase(i_2)\\
&hasBase(i_1, i_2)\\
&MozzarellaTopping(i_3)\\
&TomatoTopping(i_4)\\
&hasTopping(i_1, i_3)\\
&hasTopping(i_1, i_4)\ .
\end{aligned}
$$

These ABoxes that have been created from the paths in PathViz, can now be passed on to SuperModel to be explored further.

6.7 Conclusion

In this chapter we discussed PathViz as an implementation of CRA. PathViz visualises concept relationships in an ontology through paths. We can assign specific meanings to some of these paths and these meanings can be expressed as DL sentences. Finally, we gave an example of how two ontology understanding techniques can work together to enhance comprehension.

Seven

Measurement Instruments

7.1 Introduction

The purpose of the measurement instruments is to give an indication of the effect that entailments have on concept relationships. In this chapter, we define two measurement instruments, namely, the Path Cardinality Ratio (Section 7.2) and the Path Simplicity Ratio (Section 7.3). These two measurement instruments can be seen as ways to interpret the effect that entailments have on concept relationships. These measurement instruments were derived through the process of using and analysing the PathViz plug-in as described in Chapter 6. In Section 7.4 we will show how these measurement instruments can be interpreted in tandem. Section 7.6 explains how the measurement instruments relate to the literature discussed in Chapter 4. Section 7.5 comments on the significance of these measurement instruments within the context of ontology comprehension.

This chapter addresses our third sub research question (Q_3), namely, *How can path visualisation techniques be used to comprehend the effect of the reasoner in a formal ontology?*

7.2 Path Cardinality Ratio

In this section we introduce the Path Cardinality Ratio (PCR). The PCR is an indication of the additional paths generated by the inferred graph, given a root concept and a goal concept.

Definition 7.1 *Path Cardinality Ratio:*

$$PCR(v_r, v_g) := \begin{cases} \frac{|P_a|}{|P_i|}, \textit{with } PCR \in [0,1] & \textit{if } |P_i| \neq 0 \\ 1.0 & \textit{otherwise} \end{cases}.$$

□

Here, v_r denotes the root concept and v_g denotes the goal concept. Furthermore, $|P_a|$ indicates the number of paths found in the asserted graph between the root concept v_r and the goal concept v_g (cardinality of P_a). Similarly, $|P_i|$ indicates the number of paths found in the inferred graph (cardinality of P_i). We also note that $|P_i| \geq |P_a|$, meaning that the cardinality of the set of paths computed for the inferred graph will always be greater or equal to the cardinality of the set of paths computed for the asserted graph. This is because the ontology reasoner infers additional information that can increase the number of paths computed for the inferred graph.

From a PCR of one or close to one we can conclude that the inferred graph produced only a few more paths than the asserted graph between v_r and v_g. On the other hand, a PCR of close to zero indicates that the inferred graph produced many more paths. The value of the PCR will always lie between zero and one.

The PCR is not necessarily the same if the root and the goal concepts are switched:

$$PCR(v_r, v_g) \neq PCR(v_g, v_r) \ .$$

This is because concepts in an ontology do not necessarily have bi-directional relationships and therefore the asserted and inferred graphs are not necessarily bi-directional.

We illustrate the use of the PCR from the well-known pizza ontology [3]:

Example 7.1 *In this example we chose* `American[Pizza]` *as the root concept and* `PizzaBase` *as the goal concept from the pizza ontology. The PCR of 0.5 indicates that the inferred graph produced twice as many paths as the asserted graph. Figure 7.1 illustrates this, where the blue arrows indicate the additional paths in the inferred graph.*

$$\begin{aligned} \textit{Let } v_r = \textit{American[Pizza] and } v_g &= \textit{PizzaBase}: \\ |P_a| &= 3 \\ |P_i| &= 6 \\ PCR(American, PizzaBase) &= \frac{3}{6} = 0.5 \ . \end{aligned}$$

□

The PCR is useful in seeing how many additional paths there are between a root and goal concept in the inferred graph, but that does not necessarily give an accurate indication of the effect that the reasoner has on concept relationships in an ontology. We explain this by means of another example:

Example 7.2 *In this example we can see that the inferred graph did not produce any additional paths and intuitively the conclusion is that the reasoner does not have any effect on the way* `Margherita[Pizza]` *and* `PizzaTopping` *relate to each other.*

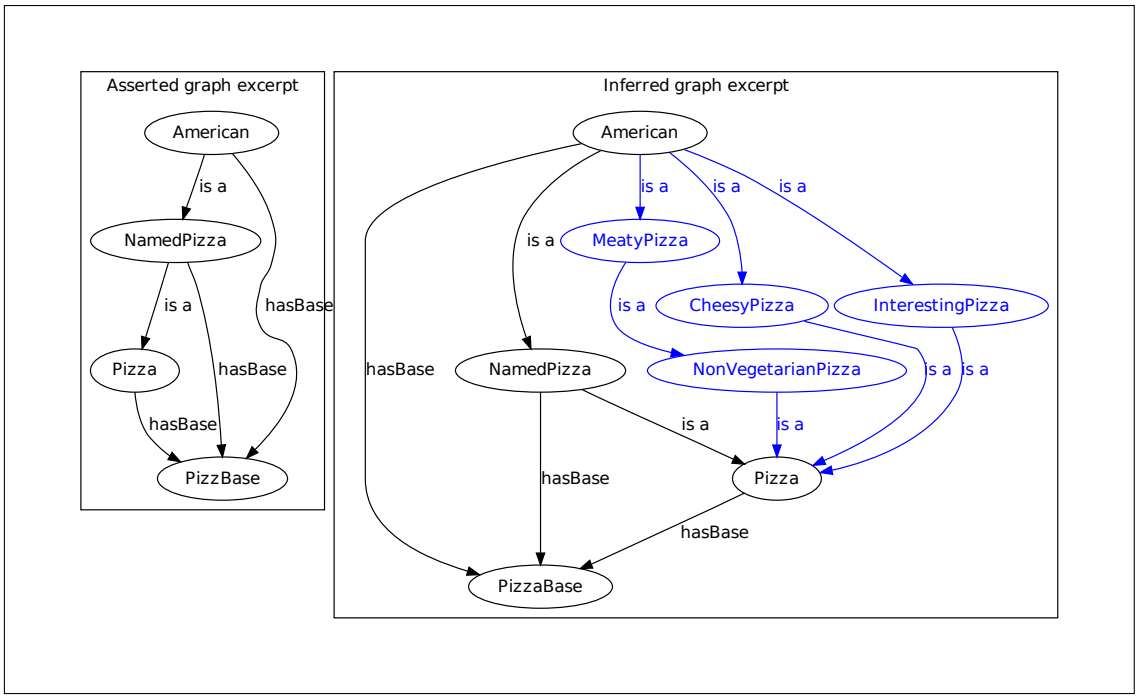


Figure 7.1: $v_r = American[Pizza]$ and $v_g = PizzaBase$

This, however, is not true. Consider Figure 7.2. Note that the paths contained in P_a and in P_i are different. The reasoner changed the structure of the taxonomy.

$$\begin{aligned} \text{Let } v_r = Margherita[Pizza] \text{ and } v_g &= PizzaTopping: \\ |P_a| &= 2 \\ |P_i| &= 2 \\ PCR(Margherita, PizzaTopping) &= \frac{2}{2} = 1.0 \ . \end{aligned}$$

□

The PCR is a good indication of the additional paths generated in the inferred graph, but it does not give a complete indication of the effect of the reasoner on concept relationships, as was shown in Example 7.2. We address this shortcoming by defining the Path Simplicity Ratio (PSR) as a second measurement instrument.

7.3 Path Simplicity Ratio

In this section we introduce the Path Simplicity Ratio (PSR). The PSR gives an indication of the overall effect that the reasoner has on the way a root and goal concept relate to each other.

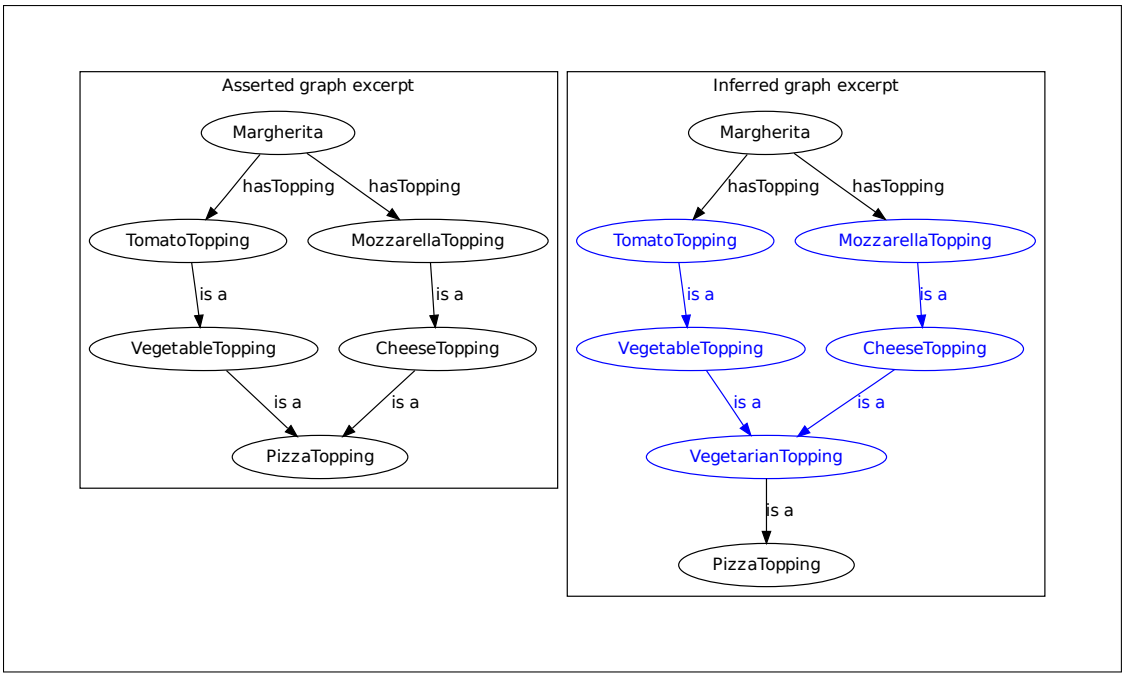


Figure 7.2: $v_r = Margherita[Pizza]$ and $v_g = PizzaTopping$

Definition 7.2 *Path Simplicity Ratio:*

$$PSR(v_r, v_g) := \begin{cases} \frac{|P_i| - |P_a \cap P_i|}{|P_i|}, \textit{with } PSR \in [0,1] & \textit{if } |P_i| \neq 0 \\ 0.0 & \textit{otherwise} \end{cases}.$$

□

Here, we subtract $|P_a \cap P_i|$ from $|P_i|$ before dividing by P_i. This ensures that *all* paths in P_i that are dissimilar to the paths in P_a are taken into account when calculating the effect of the reasoner on concept relationships.

A PSR of one means that the reasoner has a substantial effect on the way the root and goal concept relate to each other (none of the paths in the asserted graph exist in the inferred graph). A PSR of zero will mean that the reasoner has no effect on the way the root and goal concept relate to each other (all the paths in the asserted graph were found in the inferred graph and no new paths were produced in the inferred graph). The value of the PSR will always lie between zero and one. As an illustration, we extend the previous example by calculating the PSR:

Example 7.3 *In this example the PSR is one, which indicates that the reasoner had a substantial effect on the way two concepts relate to each other.*

$$\begin{aligned} Let\ v_r = Margherita\ and\ v_g &= PizzaTopping\ . \\ |P_i| - |P_a \cap P_i| &= 2 - 0 \\ |P_i| &= 2 \\ PSR(Margherita, PizzaTopping) &= \frac{2}{2} = 1.0\ . \end{aligned}$$

□

7.4 Interpretations of measurement instruments

The PCR and the PSR can be used in tandem in assessing the effect of the reasoner on concept relationships in an ontology. Table 7.1 explains how these ratios should be interpreted in tandem. All entries in Table 7.1 assume $|P_i| \neq 0$.

	Values of PCR and PSR	**Interpretation**
1	$PCR = 1.0$ and $PSR = 1.0$	No additional inferred paths ($\|P_a\| = \|P_i\|$) and no path in P_a is also in P_i ($P_a \cap P_i = \varnothing$).
2	$PCR = 1.0$ and $PSR = 0.0$	No additional inferred paths ($\|P_a\| = \|P_i\|$) and all paths in P_a are also in P_i ($P_a \cap P_i = P_i$).
3	$PCR = 0.0$ and $PSR = 1.0$	No paths in P_a ($\|P_a\| = 0$).
4	$PCR + PSR = 1.0$	There may be additional inferred paths, but all paths in P_a are also in P_i ($P_a \cap P_i = P_a$).

Table 7.1: Measurement ratio interpretation

We continue to show mathematically why the statements in Table 7.1 are true, starting with the statement that we want to prove and the information derived from the interpretation.

(1) If $|P_a| = |P_i|$ and $P_a \cap P_i = \varnothing$, then $PCR = 1.0$ and $PSR = 1.0$.

Proof:
In Definition 7.1,

$$PCR = \frac{|P_a|}{|P_i|}\ .$$

From $|P_a| = |P_i|$, it follows that

$$PCR = \frac{|P_i|}{|P_i|} = 1.0\ .$$

In Definition 7.2,

$$PSR = \frac{|P_i| - |P_a \cap P_i|}{|P_i|} \ .$$

From $P_a \cap P_i = \varnothing$, it follows that

$$PSR = \frac{|P_i| - |\varnothing|}{|P_i|} = \frac{|P_i| - 0}{|P_i|} = 1.0 \ .$$

□

(2) If $|P_a| = |P_i|$ and $P_a \cap P_i = P_i$, then $PCR = 1.0$ and $PSR = 0.0$.

Proof:

In Definition 7.1,

$$PCR = \frac{|P_a|}{|P_i|} \ .$$

From $|P_a| = |P_i|$, it follows that

$$PCR = \frac{|P_i|}{|P_i|} = 1.0 \ .$$

In Definition 7.2,

$$PSR = \frac{|P_i| - |P_a \cap P_i|}{|P_i|} \ .$$

From $P_a \cap P_i = P_i$, it follows that

$$PSR = \frac{|P_i| - |P_i|}{|P_i|} = \frac{0}{|P_i|} = 0.0 \ .$$

□

(3) If $|P_a| = 0$, then $PCR = 0.0$ and $PSR = 1.0$.

Proof:

In Definition 7.1,

$$PCR = \frac{|P_a|}{|P_i|} \ .$$

From $|P_a| = 0$, it follows that

$$PCR = \frac{0}{|P_i|} = 0.0 \ .$$

In Definition 7.2,

$$PSR = \frac{|P_i| - |P_a \cap P_i|}{|P_i|} \ .$$

From $|P_a| = 0$, it follows that

$$PSR = \frac{|P_i| - |\varnothing \cap P_i|}{|P_i|} .$$

$$\therefore PSR = \frac{|P_i| - 0}{|P_i|} = 1.0 .$$

□

(4) If $P_a \cap P_i = P_a$, then $PCR + PSR = 1.0$.

Proof:

From Definition 7.1 and Definition 7.2,

$$PCR + PSR = \frac{|P_a|}{|P_i|} + \frac{|P_i| - |P_a \cap P_i|}{|P_i|} .$$

From $P_a \cap P_i = P_a$, it follows that

$$PCR + PSR = \frac{|P_a|}{|P_i|} + \frac{|P_i| - |P_a|}{|P_i|}$$

$$= \frac{|P_a| + |P_i| - |P_a|}{|P_i|}$$

$$= \frac{|P_i|}{|P_i|} = 1.0 .$$

□

7.5 Significance of the measurement instruments

The measurement instruments provide a quantitative indication of the influence that entailments have on concept relationships. As shown in the previous section, the values obtained from these instruments have different meanings. Relationships amongst concepts in an ontology differ according to the values of the PCR and PSR. Figure 7.3 shows how these measurement instruments fit into the ontology comprehension framework.

More ways in which these instruments can be employed usefully is a topic for future research. As an example, concept relationships that are influenced by entailments can be extracted from the ontology and employed in ontology learning strategies to see whether emphasising those concepts accelerates understanding.

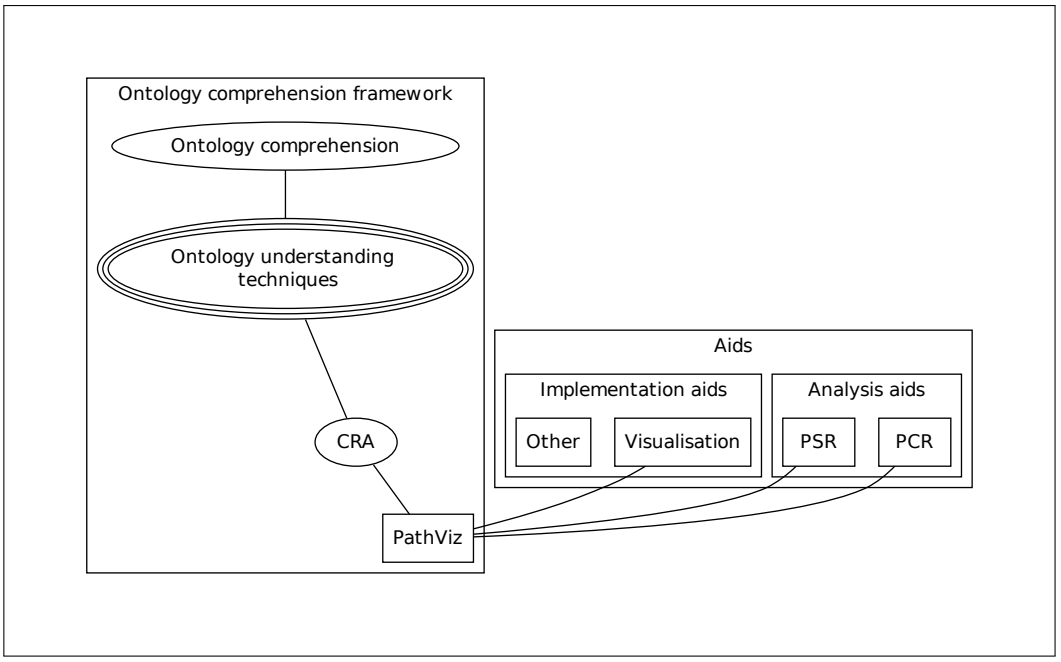


Figure 7.3: Analysis aids in PathViz

7.6 Measurement instruments and ontology debugging

Section 4.3 in Chapter 4 described literature that studied the effect of entailments in the context of ontology debugging. Here, the focus was on giving adequate explanations for the classification of unsatisfiable classes by the ontology reasoner. The aim is to help the ontology developer understand why the reasoner classified certain classes as unsatisfiable so that an appropriate course of action can be taken to repair the broken ontology.

The measurement instruments as described in this chapter differ from ontology debugging techniques. Even though both study the effect of entailments, the focus of the measurement instruments is not to give reasons for decisions made by the ontology reasoner. The focus is on showing how entailments affect the relationships that concepts in an ontology have with each other.

7.7 Conclusion

In this chapter we described a formal way to comprehend the effect that entailments have on concept relationships. To achieve this, two measurement instruments, the PCR and PSR, were defined. Additionally, we illustrated how the values of these instruments can be interpreted. Finally, the measurement instruments were

compared with ontology debugging techniques described earlier.

The next chapter, Chapter 8, serves as an evaluation that aims to validate the answers that have been given to the research questions.

Eight

Evaluation

8.1 Introduction

In this chapter we conduct an evaluation of PathViz. The purpose of this evaluation is to validate the answers to the research questions in previous chapters.

The evaluation of PathViz consists of three parts:

- Firstly, we evaluate PathViz with the ontology visualisation framework as described in Chapter 3 that was used on a selection of ontology visualisation software tools (more details in Section 8.2).
- In the second part of the evaluation, we develop a survey that was consequently completed by a selected audience. More details surrounding the survey can be found in Section 8.3. The results of the survey are discussed in Section 8.4.
- In the last part of the evaluation we give participants a task to do in Protégé without PathViz and a task to do with PathViz (Section 8.5). We conclude in Section 8.7.

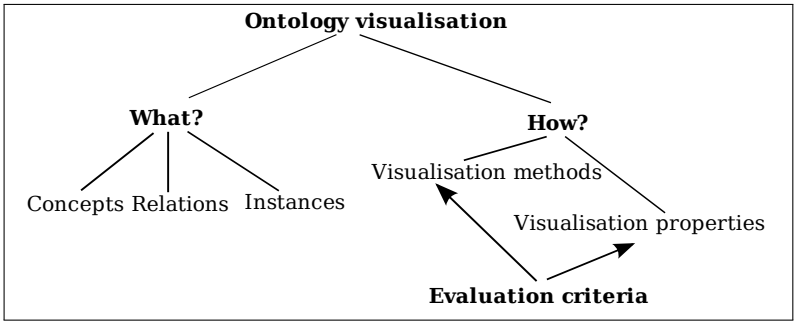


Figure 8.1: Ontology visualisation framework

8.2 Ontology visualisation framework

In Chapter 3 we discussed a selection of ontology visualisation tools in an ontology visualisation framework (Figure 8.1). In this section we continue that discussion by describing PathViz within that framework. Although this is strictly speaking not an evaluation with measurable results, we do regard it as an descriptive evaluation and we use it to round off the literature analysis done in Chapter 3.

OWLViz	TGVizTab	OntoSphere	Swoop	ClusterMap	VantagePoint	SuperModel and PathViz

Table 8.1: Selection of ontology visualisation tools for evaluation

PathViz is a two-dimensional Protégé plug-in that visualises concept relationships in an ontology. It also visualises the influence that entailments have on concept relationships. PathViz uses a diagonal path visualisation method to illustrate concept relationships. The path that illustrates the relationships between two concepts start in the top left hand corner of the screen and progressively moves down to the bottom right hand corner of the screen. PathViz uses shape, colour and opacity as visualisation properties. Concepts appear inside circles and these circles are connected to each other with arrows. Paths in the inferred graph that are not found in the asserted graph are highlighted with blue to indicate that the reasoner had an effect on the way the two concepts relate to each other. PathViz uses opacity to indicate progression in the paths. The root node has a light grey background and the nodes become progressively darker as they come closer to the goal node. Small variations in the opacity from root node to goal node indicates that the path is long, while large variations are indicative of a short path. For example, a path with two nodes (root node directly connected with goal node) will have one light grey node and one dark grey node. Table 8.2 extends the summary of ontology visualisation tools of Chapter 3 and now includes SuperModel and PathViz.

8.3 Survey

The evaluation of PathViz in an ontology visualisation context is useful, but it still does not help us determine whether the ontology understanding technique (CRA) implemented in PathViz enhances comprehension. To achieve this, another evaluation method is required. PathViz was demonstrated to an audience familiar with ontologies and they filled out a survey after the demonstration.

It is important to ask the right questions in the survey to test whether the goals of PathViz were met. We identify three important goals that PathViz should meet:

Software tool	Visualisation method	Visualisation properties
OWLViz	tree layout	colour, shape, 2-dimensional
TGVizTab	spring layout	colour, 2-dimensional
OntoSphere	hyperbolic tree layout	colour, size, 3-dimensional
Swoop	cropcircles	colour, size, 2-dimensional
ClusterMap	spring layout	colour, opacity, 2-dimensional
VantagePoint	model specific	model specific, 3-dimensional
SuperModel	spring layout	shape, colour, 2-dimensional
PathViz	diagonal layout	shape, colour, opacity, 2-dimensional

Table 8.2: Summary of ontology visualisation tools

- PathViz should have helped users to better understand ontologies.
- PathViz should have helped users to understand how different concepts in the ontology are related to each other.
- PathViz should have helped users to understand what influence the reasoner had on the way concepts in the ontology are related to each other.

These goals correspond to the research questions that were initially posed in Chapter 1, namely:

- How can the use of path visualization techniques applied to subsumption and existential relationships between concepts in an ontology, enhance ontology comprehension within an ontology comprehension framework?
- How can we construct an ontology comprehension framework wherein we can classify the approaches related to ontology comprehension?
- How can we apply path visualisation techniques to comprehend subsumption and existential relationships between concepts in an ontology?
- How can path visualisation techniques be used to comprehend the effect of the reasoner in a formal ontology?

The questions in the survey can be found in Table 8.3, Table 8.4 and Table 8.5. The questions were divided into three sections: comprehension, usability, definitions and measurement instruments. The questions under the *comprehension* and *definitions and measurement instruments* sections were specifically designed to test whether the goals of PathViz have been met. The other section focuses on the usability of PathViz. Questions were answered on a Likert scale (1–5) where 1 indicates that the participant strongly disagrees and 5 indicates that the participant strongly agrees with the statement. The final section of the survey provided participants with an opportunity to give additional comments and suggestions about

Comprehension
PathViz facilitates my understanding of an ontology
PathViz helps me to understand how two concepts in an ontology are related to each other
PathViz helps me to understand what effect the ontology reasoner has on the way concepts in the ontology relate to each other

Table 8.3: Survey–Comprehension

Usability
PathViz is easy to use
PathViz makes effective use of colours to highlight relevant information
PathViz is easy to navigate

Table 8.4: Survey–Usability

Definitions and measurement instruments
An ontology comprehension framework is a useful mechanism to describe and categorise software tools that aid ontology comprehension
The *Path Cardinality Ratio* is a useful measurement instrument for indicating the influence the ontology reasoner has on the way concepts in the ontology relate to each other
The *Path Simplicity Ratio* is a useful measurement instrument for indicating the influence the ontology reasoner has on the way concepts in the ontology relate to each other

Table 8.5: Survey–Definitions and measurement instruments

their experience. Participants that were not familiar with Protégé, attended a two day workshop beforehand. Prior to filling out the survey, all the participants attended a talk on ontology comprehension and a demonstration of PathViz. Eleven participants completed the survey.

8.4 Survey Results

8.4.1 Introduction

The survey was divided into three categories and we proceed with the discussion of the results under those three categories. These three categories tested the goals we wanted to meet with PathViz as well as the usability of the software. Section 8.4.5 elaborates on additional comments given by participants.

8.4.2 Comprehension

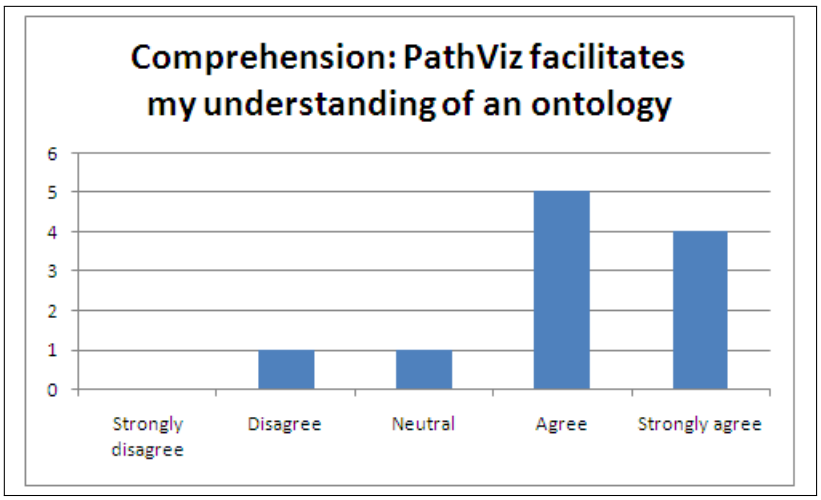


Figure 8.2: Comprehension – Question 1

In the first question participants were asked whether PathViz helped them to understand an ontology. Results for this question can be found in Figure 8.2. In this question the vast majority of participants indicated that they agreed or strongly agreed. In other words, PathViz was well-received as a tool that aids the general comprehension of an ontology.

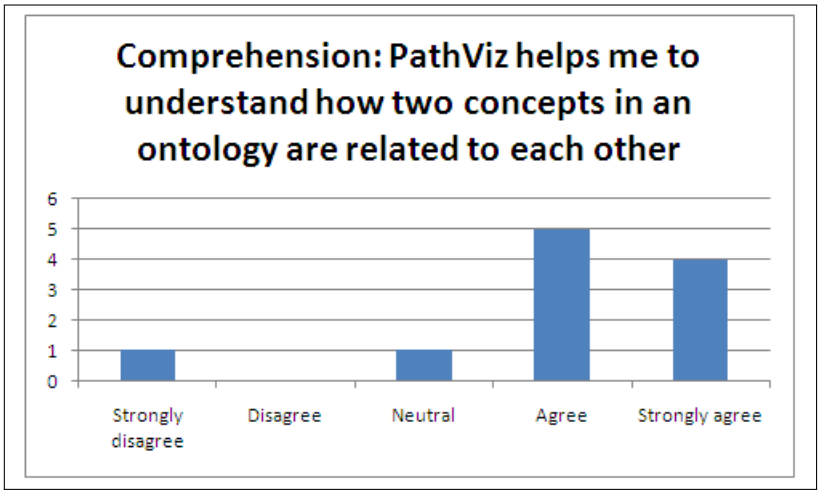


Figure 8.3: Comprehension – Question 2

The second question asked participants to evaluate a more specific aspect related to the comprehension of an ontology. They were asked whether PathViz helped

them to understand how two concepts in an ontology relate to each other. Results were similar to those obtained in the first question with most participants agreeing or strongly agreeing (Figure 8.3). We interpret this as PathViz being a successful implementation of CRA as an ontology understanding technique.

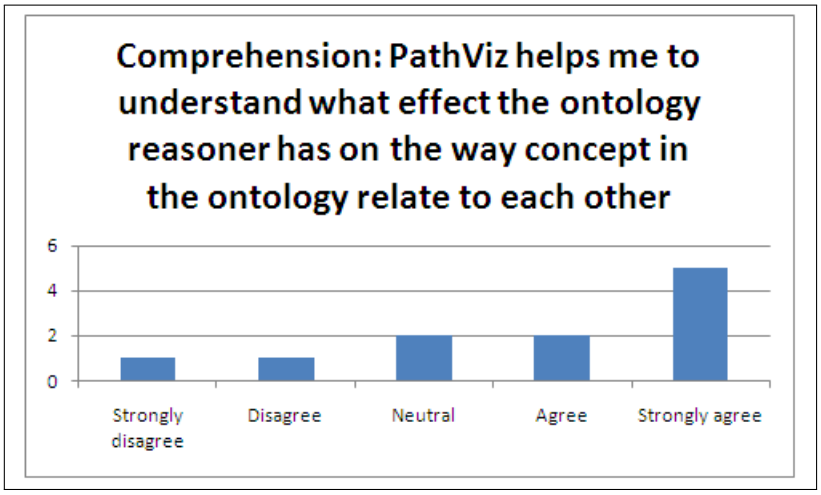


Figure 8.4: Comprehension – Question 3

In the third question the focus shifted to the influence of the ontology reasoner. Participants were asked if PathViz helped them to understand the effect that the ontology reasoner (entailments) had on concept relationships. Results for this question were favourable, but not as favourable as for the first two questions. Participants commented that they would have liked more time to evaluate PathViz in order to understand the influence of the reasoner. In addition, a complete grasp of the influence of the reasoner requires a good understanding of the PCR and PSR. Therefore, we do not regard the large portion of neutral answers as something negative. Given more time in their evaluation of PathViz, more participants might have responded more positively to this question.

8.4.3 Usability

During the implementation of PathViz, certain aspects, such as the use of colour and easy navigation, were designed to improve the usability of the software. We argue that more usable software accelerates comprehension. In all three questions, participants responded either neutrally or positively. Participants found the step-based wizard approach for improved navigation especially useful.

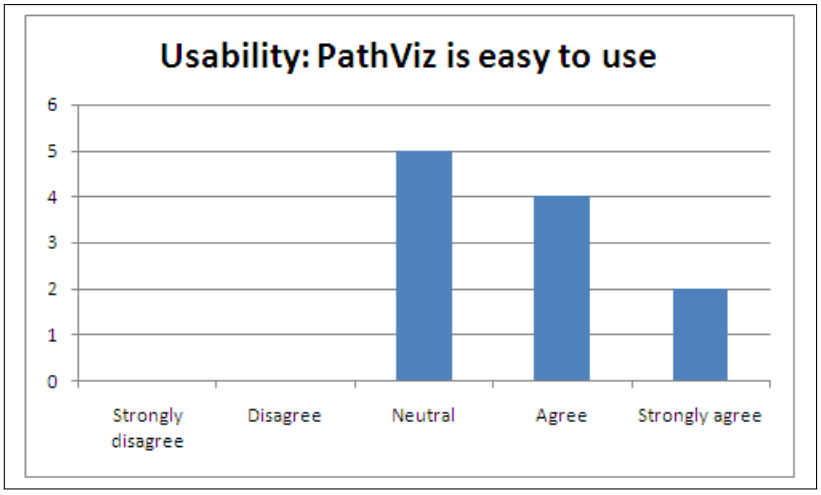


Figure 8.5: Usability – Question 1

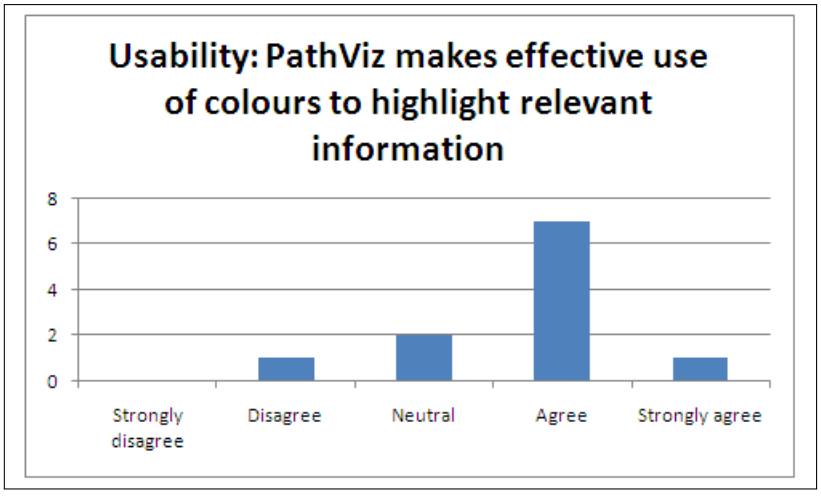


Figure 8.6: Usability – Question 2

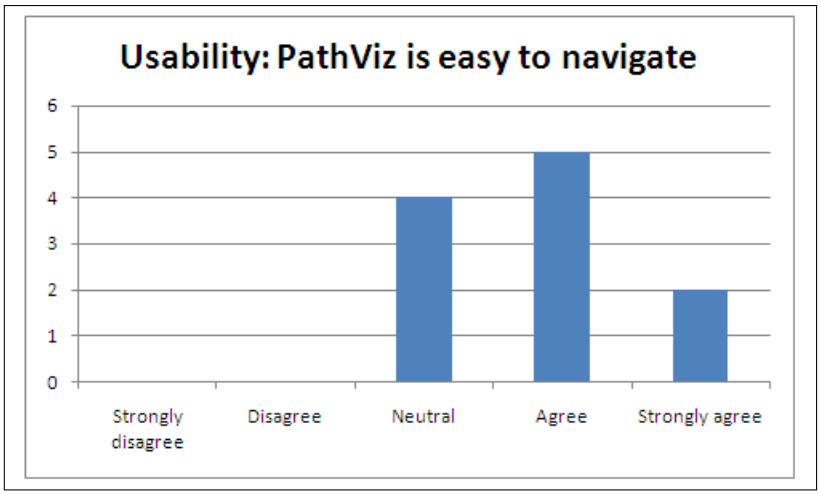


Figure 8.7: Usability – Question 3

8.4.4 Definitions

Questions in this section centered around the ontology comprehension framework and the measurement instruments. Again, participants responded favourably. Most participants were neutral or agreed that the ontology comprehension framework is a useful mechanism for describing and categorising software tools that aid comprehension. Most participants also found the measurement instruments useful indicators with regard to the influence of the ontology reasoner (entailments) on concept relationships.

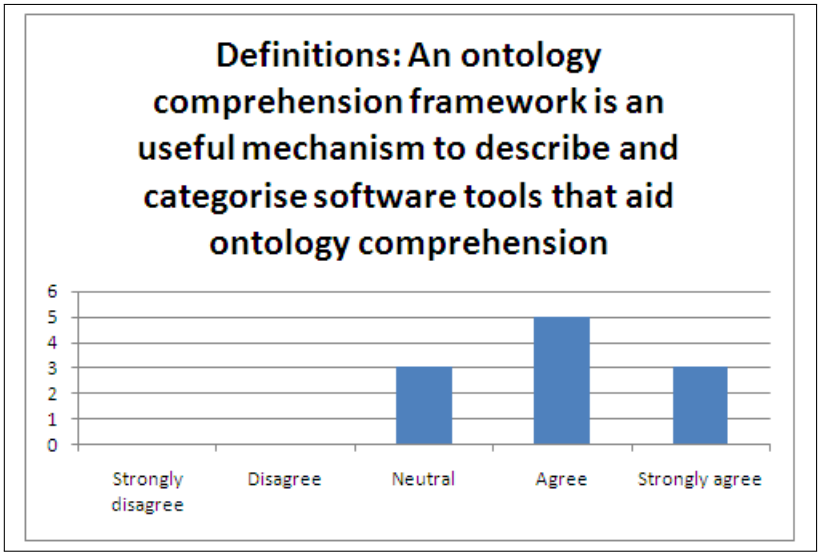


Figure 8.8: Definitions – Question 1

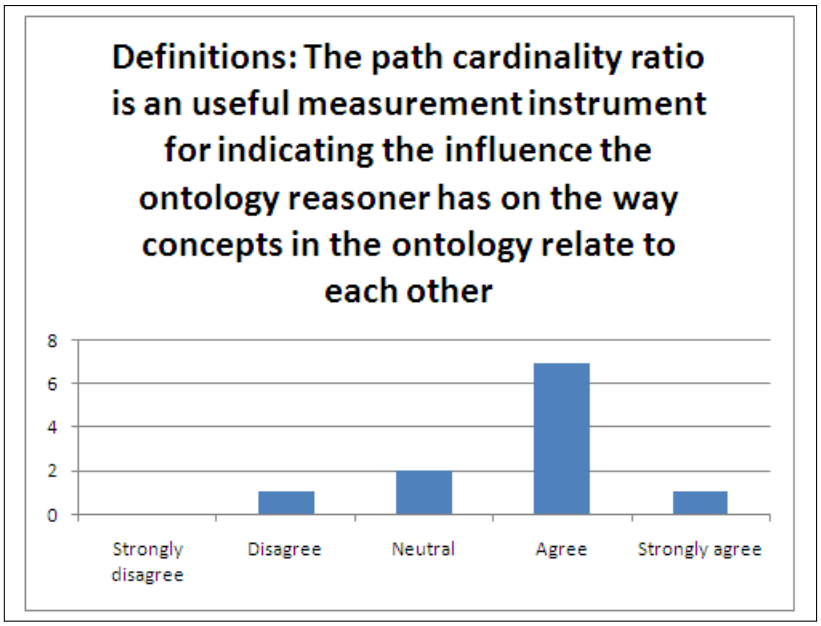


Figure 8.9: Definitions – Question 2

8.4.5 Other comments

In general, participants commented that PathViz is a useful tool. They also acknowledged the importance of having software support for ontology comprehension.

Some of the more specific comments were the following:

- Participants requested more time to evaluate PathViz to understand how PathViz shows the influence the ontology reasoner has on concept relationships.
- It can be useful to place stronger emphasis on the difference between subsumption and existential relationships in the paths.
- Consider existing graphical formalisms such as Entity Relationship diagrams and UML for future work.
- Another measurement instrument in addition to the PCR and PSR could use the length of the paths to indicate the influence the ontology reasoner has on concept relationships.

The difference between subsumption and existential relationships can be highlighted by using distinct colours for these relationships. Using the length of the paths in PathViz to indicate the influence of the ontology reasoner (entailments) is

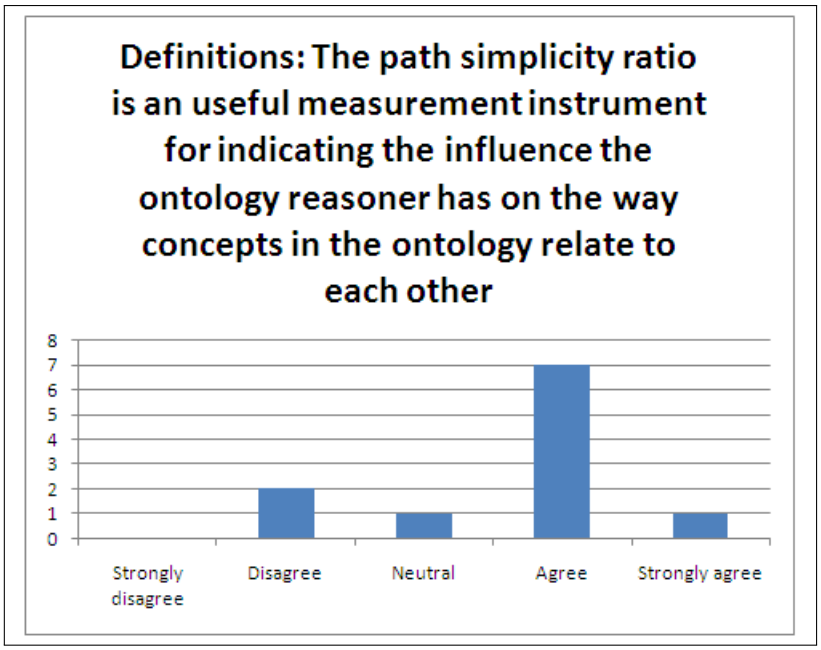


Figure 8.10: Definitions – Question 3

an easy calculation. This can be done by comparing the number of nodes in the paths computed by the assserted and inferred graphs respectively.

8.5 Task-based evaluation in Protégé

In this part of the evaluation participants were given two tasks to complete in Protégé. These tasks are shown in Table 8.6 and Table 8.7. The participants completed the first task by using the Protégé ontology editor. The second task was similar to the first task, but users had to use the PathViz plug-in. Finally, participants were asked which of the two tasks were easier to complete.

We experienced a number of difficulties with this evaluation. Firstly, many of the participants did not have laptops and some struggled to install the PathViz plug-in. The end result was that participants worked in groups and only three surveys were returned.

From the surveys that were returned, we could not conclusively say that the second task was easier to complete than the first task. However, participants did indicate that PathViz gave them additional information regarding concept relationships that they would not otherwise have known. A better approach for the task based evaluation would have been to give tasks that have a definitive correct answer.

Task 1
Consider the well-known pizza ontology. Use the *Protégé ontology editor* and (1) describe the relationship between the initial concept `Capriana[Pizza]` and the goal concept `PizzaBase`. (2) What effect does the ontology reasoner have on the way these concepts relate to each other?

Table 8.6: Task 1

Task 2
Consider the well-known pizza ontology. Use the *PathViz Protégé plug-in* and (1) describe the relationship between the initial concept `Soho[Pizza]` and the goal concept `PizzaBase`. (2) What effect does the ontology reasoner have on the way these concepts relate to each other? *Remember to execute the reasoner (e.g. Fact++) in Protégé, before using PathViz*

Table 8.7: Task 2

8.6 Threat to validity

Concepts of validity have an substantial impact on how researchers think about their work [20]. Factors that threaten the validity of research results can decrease the significance of the results. Several authors (for example [20, 45]) have identified various factors that can threaten the validity of research. These factors are placed in different categories. For example, Calder *et al.* [20] classify these factors in two categories (*internal validity* and *external validity*), while Singleton *et al.* [45] classify them in four categories (*internal validity*, *external validity*, *statistical conclusion validity* and *construct validity*). For our purposes, we interpret *threat to validity* to be any factor that could decrease the significance of the research results.

In consideration of the above-mentioned results, two possible threats were identified:

- Sampling bias
- Acquiescence bias

The group of participants that completed the survey was very small (only eleven). Therefore, it is reasonable to argue that such a small sample decreases the significance of the results. However, although the sample was small, it was representative in the sense that it included people from different nationalities, races and gender.

Acquiescence bias occurs when questions are posed in such a way that respondents are more prone to give certain answers. For example, when questions inherently make positive statements about a product, respondents are more likely to give a positive response about the product. The questions in the survey could

possibly be interpreted as having an acquiescence bias. Even though the questions where answered by means of a Likert scale, the questions inherently stated that the ontology comprehension framework, PathViz and the measurement instruments aid comprehension.

Not all participants had the same level of knowledge about ontologies. Therefore, it is debatable whether a sample of experts would have responded the same to the survey.

8.7 Conclusion

This chapter discussed an evaluation of PathViz. Firstly, we did a descriptive evaluation of PathViz by describing it within the ontology visualisation framework of Chapter 3.

In the second part of the evaluation we designed a survey to test whether the goals of PathViz were met. Participants responded favourable to all questions, but noted that they wanted more time to use PathViz to understand the influence of the reasoner on concept relationships.

In the last part of the evaluation we used a task-based approach to evaluate PathViz. We encountered problems during this evaluation and it was not as successful as hoped. However, participants indicated that PathViz gave them additional information regarding concept relationships in an ontology.

Finally, when referring to the main research question, the results of our evaluation indicate that the use of path visualisation to analyse the relationships between concepts in an ontology does enhance the comprehension of an ontology.

Nine

Conclusion

9.1 Summary

The use of ontologies are becoming more widespread and therefore the need also arose for quicker and better comprehension. In the background chapters we investigated a selection of ontology visualisation tools. We found that certain ontology visualisation tools visualise the stated information in the ontology, while others actually visualise much more. They visualise some ontology understanding technique. Definition 5.2 in Chapter 5 described what an ontology understanding technique is.

The identification of ontology understanding techniques during the visualisation proses was an important step in the consequent formulation of an ontology comprehension framework. We defined our ontology comprehension framework to be a collection of all the ontology understanding techniques (Definition 5.1 in Chapter 5). These ontology understanding techniques in turn can interact with each other to enhance comprehension. Each of these ontology understanding techniques can have one or more concrete implementations. We continued to classify model exploration as an ontology understanding technique and SuperModel as a specific implementation of it.

We then developed a new ontology understanding technique, Concept Relationship Analysis (CRA). CRA was defined as the ontology understanding technique that analyses the relationships between concepts in the ontology (Definition 5.3 in Chapter 5). Consequently, we implemented a Protégé plug-in, PathViz, that was a specific implementation of CRA. PathViz builds two graphs (asserted and inferred) of the ontology by using subsumption and existential relationships. Paths taken from these graphs show how two concepts in the ontology are related to each other. Technical details on the PathViz implementation were provided in Appendix B. A Background on Protégé and plug-in development in Protégé were provided in Appendix A.

We continued by analysing the information obtained from these graphs and con-

sequently defined two measurement instruments. These measurement instruments, PCR (Definition 7.1) and PSR (Definition 7.2), are mathematical ratios that help us to understand the influence that entailments have on concept relationships in an ontology.

Answers provided to the research questions in Chapter 5, Chapter 6 and Chapter 7 were validated by means of a survey in Chapter 8. Participants in this survey responded favourably to questions posed on the ontology comprehension framework, PathViz and the measurement instruments.

9.2 Contributions

With this research, we made the following contributions:

- An ontology comprehension framework wherein we can discuss ontology understanding techniques and implementations of these ontology understanding techniques
- The development of Concept Relationship Analysis (CRA) as an ontology understanding technique
- PathViz as a Protégé plug-in that implements the CRA ontology understanding technique
- Path Cardinality Ratio (PCR) and Path Simplicity Ratio (PSR) as measurement instruments that help us to understand the influence of entailments on concept relationships

9.3 Research questions revisited

In this section, we reflect on the research questions that were initially posed and explain how they were answered.

> *Q_1: How can we construct an ontology comprehension framework wherein we can classify the approaches related to ontology comprehension?*

An ontology comprehension framework can be constructed by investigating existing comprehension aids. The lessons learnt from existing comprehension aids can be abstracted an applied to ontologies. In this study, we discovered that the notion of an ontology understanding technique is important for effective ontology comprehension. Consequently, we proceeded to define ontology comprehension as a collection of all ontology understanding techniques (Definition 5.2 in Chapter 5). Each of these ontology understanding techniques can have many concrete implementations.

> *Q_2: How can we apply path visualisation techniques to comprehend subsumption and existential relationships between concepts in an ontology?*

Firstly, we defined an ontology understanding technique, namely concept relationships analysis (CRA). CRA is an ontology understanding technique that facilitates the process of understanding ontologies by analysing the relationships amongst concepts in an ontology (Definition 5.3). Consequently, we developed PathViz, a Protégé plug-in, that is a concrete implementation of CRA. PathViz builds a graph of concepts in an ontology that are linked to each other with subsumption and existential relationships. PathViz then computes a set of paths between two selected concepts in the graph. These paths are rendered visually to the user. Therefore, it is reasonable to conclude that PathViz is a software implementation that applies path visualisation techniques to facilitate the comprehension of subsumption and existential relationships between concepts in an ontology.

> *Q_3: How can path visualisation techniques be used to comprehend the effect of the reasoner in a formal ontology?*

The question will be answer by continuing our discussion on PathViz. PathViz constructs two graphs (the asserted and inferred graph). Both of these graphs connect concepts in the ontology to each other with existential and subsumption relationships, but the inferred graph takes the effect of entailments into account during graph construction. Therefore the inferred graph will typically be larger than the asserted graph, because entailments are taken into account. Consequently, we defined two measurement instruments, PCR (Definition 7.1) and PSR (Definition 7.2), that investigates the paths between two concepts in both the asserted and inferred graphs to get an indication of the effect of entailments.

Additionally, paths that occur in the inferred graph but not in the asserted graph are highlighted in a distinct colour to show users that the reasoner had an influence on the computation of these paths. Therefore, we can say that the measurement instruments defined in PathViz as well as the usage of colour to highlight inference can be used to comprehend the effect of the reasoner in a formal ontology.

> *Main research question (Q_0): How can the use of path visualization techniques applied to subsumption and existential relationships between concepts in an ontology, enhance ontology comprehension within an ontology comprehension framework?*

Finally, we return to the main research question. The answer to this question can be given by summarising the information from the three sub research questions described above. PathViz is a software implementation that makes use of path visualisation techniques. PathViz implements an ontology understanding technique, CRA, that is part of an ontology comprehension framework.

9.4 Evaluation revisited

The goal of the evaluation (Chapter 8) in the context of this study was not to obtain results in order to answer the research questions, but to validate the answers that have been provided in Chapter 5, Chapter 6 and Chapter 7. In this section, we discuss why the evaluation in Chapter 8 is proof that the research questions were answered.

The aim of the validation was to show that the artefacts that were constructed during this research were not only answers to the research questions, but were also useful to users. We involved users in the evaluation process and asked them questions about the ontology comprehension framework, PathViz and the measurement instruments. The users responded favourably to all three of the above-mentioned. Therefore, we can conclude that the artefacts that were implemented to answer the research questions were validated as successful implementations by a group of users.

9.5 Future work

The following were identifued as potential further work:

- Another measurement instrument in addition to the PCR and PSR could use the length of the paths to indicate the influence that entailments have on concept relationships.
- Find sensible ways to include universal relationships in PathViz.
- Find more practical usages of the PCR and PSR.
- An investigation into how ontology understanding techniques can be used in tandem to enhance comprehension.
- An investigation into whether the work done on ontology debugging can be classified in the ontology comprehension framework.
- Can ontology understanding techniques in the ontology comprehension framework facilitate ontology debugging?

A

Protégé

A.1 Introduction

This appendix provides more details on the Protégé ontology editor (Section A.2) and plug-in development in Protégé (Section A.3). Information in Section A.2 was derived from Horridge [30] and the usage of Protégé. Information in Section A.3 was derived from the implementation of PathViz, a Protégé plugin (also see Chapter 6 and Appendix B). Usage of documentation [4, 5, 6] assisted during the implementation of PathViz.

A.2 Protégé

In this section, we discuss the following main features in Protégé:

- creation of the ontology
- editing of the ontology
- reasoning
- the use of plug-ins

Figure A.1 displays a typical Protégé screen. Here, Protégé portrays a tabbed environment.

These tabs in Protégé correspond roughly to the elements of ontologies (ontological vocabulary) as described in Chapter 2. On each of these tabs, one section of the ontology vocabulary can be *created* or *edited*. Depending on the specific tab, concepts, roles or instances in the ontology can be edited. In this section focus will fall on four of these tabs:

- Classes

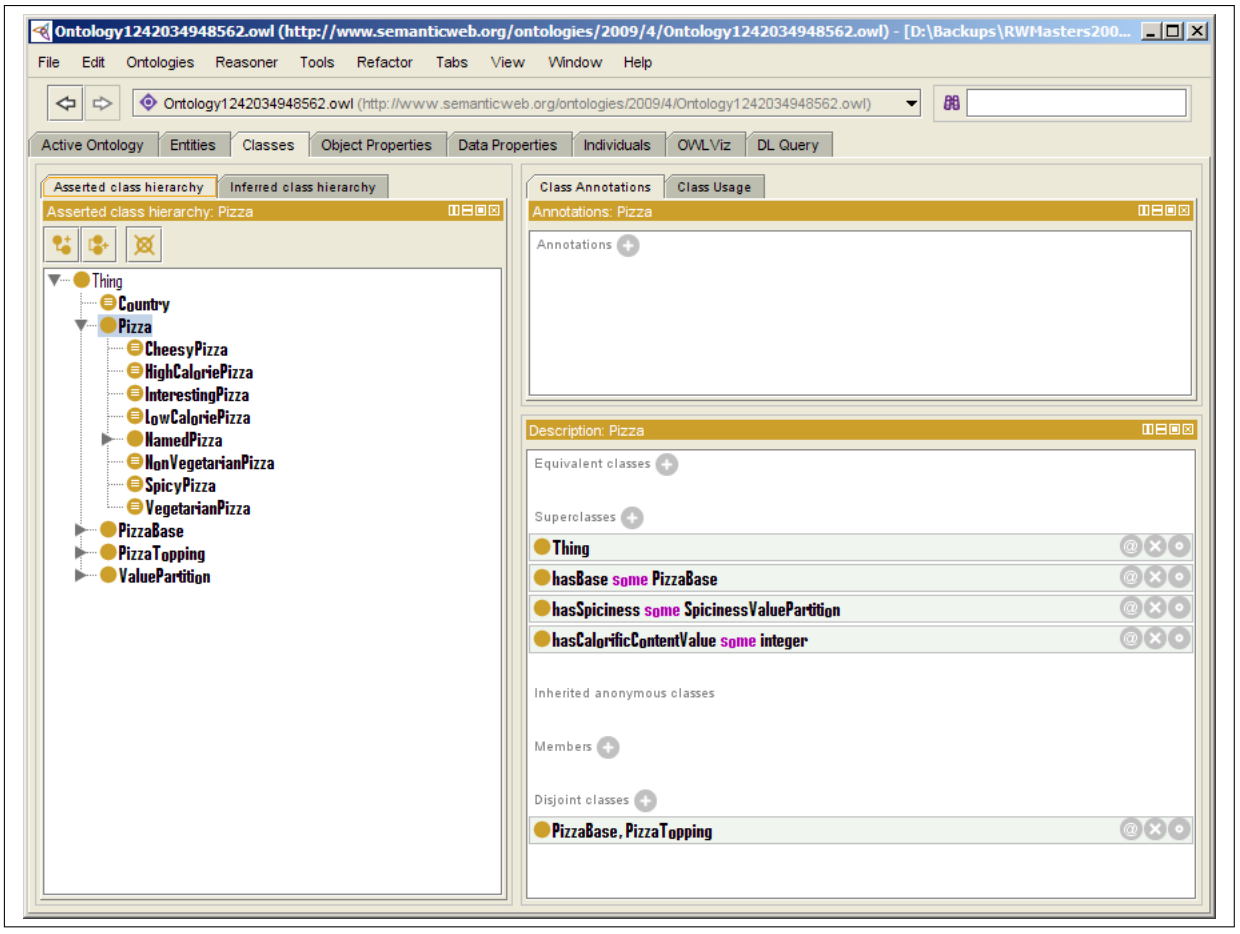


Figure A.1: Protégé ontology development tool

- Object Properties
- Data Properties
- Individuals

On the *Classes* tab, concepts are created and edited. Classes (concepts) are created and placed in a hierarchy (tree like structure as seen in Figure A.1). This is useful as it enables us to see which classes are sub-classes of other classes. This hierarchy is called the taxonomic hierarchy of the ontology. On the right hand side of the tree structure there is a description box where the user can edit the properties of the selected class. This description box has five sections:

- Equivalent classes
- Superclasses

- Inherited anonymous classes
- Members
- Disjoint classes

Equivalent classes are related to the idea of necessary and sufficient conditions in logic. It refers to the existence of other classes that will guarantee the truth (existence) of this class and that is necessary for the truth of this class. Superclasses are related to the idea of necessary conditions in logic. It refers to the existence of other classes that are necessary for the truth of this class. However, it does not guarantee the truth of this class. Inherited anonymous classes are properties that have been inherited from superclasses. These properties are not editable.

Members refer to individuals (instances) that have this class as their type. If we have an individual of this type of class then disjoint classes refer to other classes that are not simultaneously the type of this individual. Individuals can be added and edited on the *Individuals* tab in Protégé.

Next we discuss property types. Properties in Protégé refer to roles. Property types (roles) can be set up in a tree like hierarchy as is the case with classes. We distinguish between *object properties* and *data properties*. Object properties refer to roles that connect one class with another class. Object properties can take on several characteristics, such as being transitive and symmetric. Data properties refer to roles that connect one class to a defined primitive type.

We know from Chapter 4 that *reasoning* is an important task in ontologies. Reasoning enables us to infer implicit knowledge from the explicit knowledge stated in the ontology. Protégé supports reasoning and the user can choose from several reasoning engines. The user can invoke the reasoning process by selecting the reasoning option from the menu-bar. The result of the reasoning process is an inferred class hierarchy. This inferred class hierarchy is displayed next to the asserted class hierarchy on the *Classes* tab. This is useful for displaying information that has been inferred.

A.3 Protégé plug-in development

Protégé was designed so that it is easy to extend the functionality with *plug-ins* (external software components). It is important for the software developer to understand the architecture of Protégé, especially if the aim is to write a plug-in. Protégé was built on the *model-view* principle, meaning that the user-interface (view) and the program logic (model) have been separated in the program design. The view can query the model for information and send updated information to the model. The model finally writes all updates to the underlying *.owl* file.

In the remainder of the Protégé architecture discussion we will focus on the *model* part of the architecture as outlined above. Figure A.2 is a high level depiction of the Protégé software architecture. The image shows the relationship between the Protégé core system and plug-in components. When considering the Protégé core system, we again notice the model and view that were briefly discussed above. Grouped with the model part of the architecture is the *API* (Application Programming Interface). The API is a critical part of the model because it exposes important information regarding the underlying ontology. This information can be accessed and used by plug-in components.

Furthermore, the Protégé core system also exposes a set of interfaces. These interfaces must be implemented by plug-in components in order for it to be incorporated as a part of the Protégé application.

The last block in the core system is *events*. An application event is a well-known concept in software development. Application events occur, for example, when a user clicks on a button or accesses the application menu-bar. The Protégé application also has a set of important events. When these events occur, plug-in components need to be notified so that they can react accordingly. This is important because the plug-in components need to update themselves so that their state is consistent with the main application state.

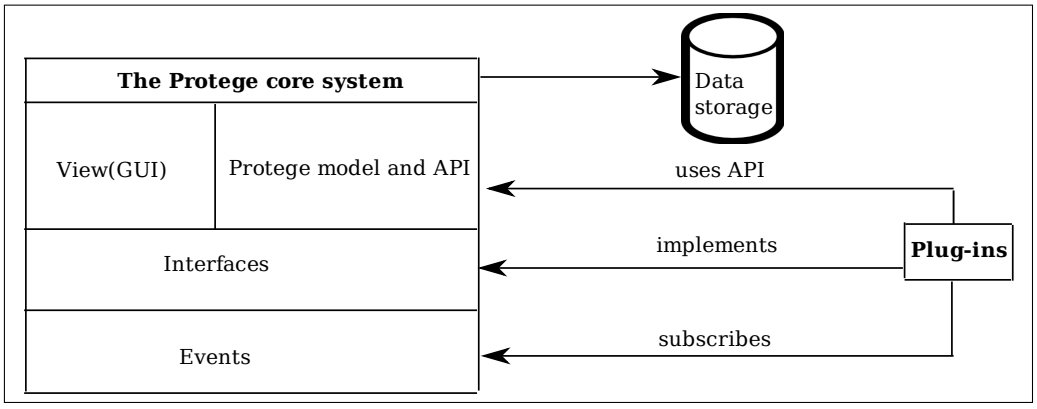


Figure A.2: Protégé software architecture

B

PathViz

B.1 Introduction

This appendix gives more detail on the PathViz Protégé plug-in. With the aid of UML diagrams more details will be given surrounding the PathViz implementation. Section B.2 deals with the PathViz software architecture and Section B.3 shows how PathViz, as a Protégé plug-in, interacts with the OWL API.

B.2 PathViz software architecture

This section describes the software architecture of the PathViz plug-in. Readers will recall that PathViz was implemented as a wizard-like tool that had three steps. Figure B.1 shows the abstract framework that was used to implement the wizard. Here, the class `CL_ApplicationProcess` can contain many classes of the type `CL_FrameBuilderProcess`. Both classes are abstract and `CL_FrameBuilderProcess` represents a step in the wizard. Each `CL_FrameBuilderProcess` can consist of many frames (widgets). These frames are represented by the class `CL_BaseFrame` (also an abstract class). In addition, there can be many data objects in each step. They are represented by the abstract class `CL_Data`.

Figure B.2 shows how this framework was used to implement PathViz. Here, `CL_PathVizApplication` is an implementation of the abstract class `CL_ApplicationProcess`. The classes `CL_FrameBuilderSelect` and `CL_FrameBuilderSVGDisplay` are implementations of the abstract class `CL_FrameBuilderProcess`. These classes represent the steps in the PathViz process. The class `CL_FrameBuilderSelect` has two instantiations, because the first two steps in the wizard is the same (recall that the user selects two concepts in the ontology in the first two steps). The class `CL_FrameBuilderSVGDisplay` is the final step in the wizard and it displays the results of the user inquiry. The classes `CL_SelectClassFrame` and `CL_DisplaySVGFrame` can be seen as the frames (widgets) that was employed by the different steps in the wizard. The class `CL_DataObjectPathSearch` is a data

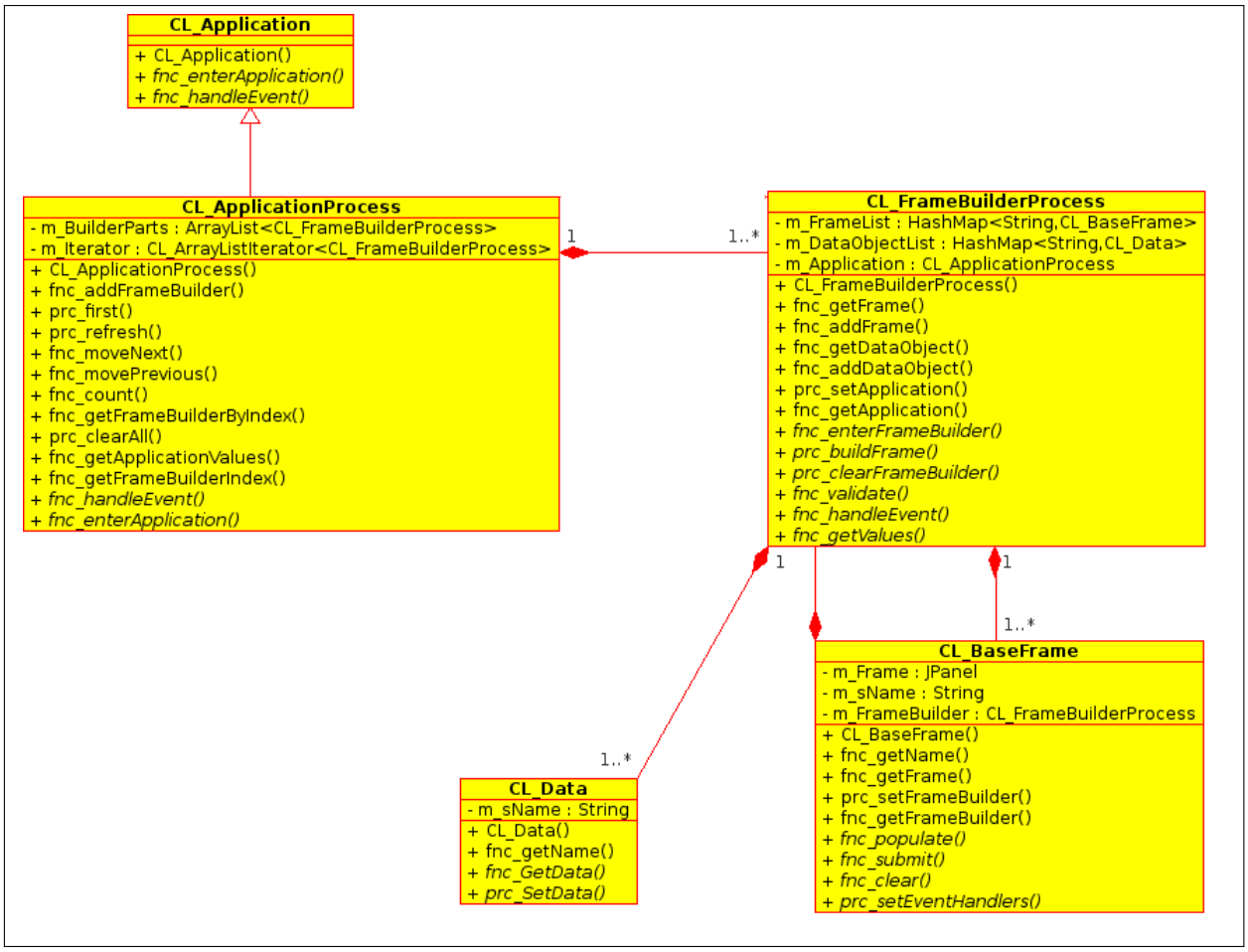


Figure B.1: Abstract navigation framework

object and it is contained in `CL_FrameBuilderSVGDisplay`. This class is used to compute the paths between two chosen concepts in an ontology. Finally, the class `CL_GraphNavigationHelper` is used to order and analyse the results obtained by `CL_DataObjectPathSearch`.

B.3 Pathviz as a Protégé plug-in

In this section, we place PathViz in the context of a Protégé plug-in. Figure B.3 shows how PathViz was implemented as a Protégé plug-in and how PathViz uses the OWL API. To realise a Protégé plug-in, the abstract class in the OWL API, `AbstractOWLClassViewComponent`, needs to be implemented. The class `PathVizView` did such an implementation. An abstract method of particular importance in the class `AbstractOWLClassViewComponent` is `updateView`. This method serves as a

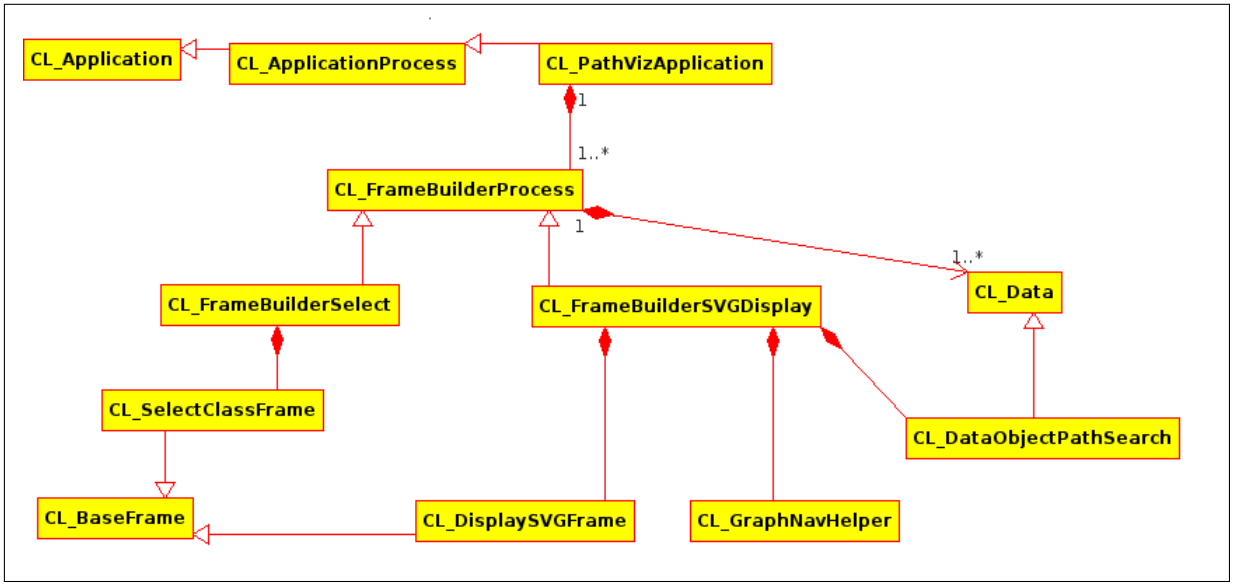


Figure B.2: PathViz implementation

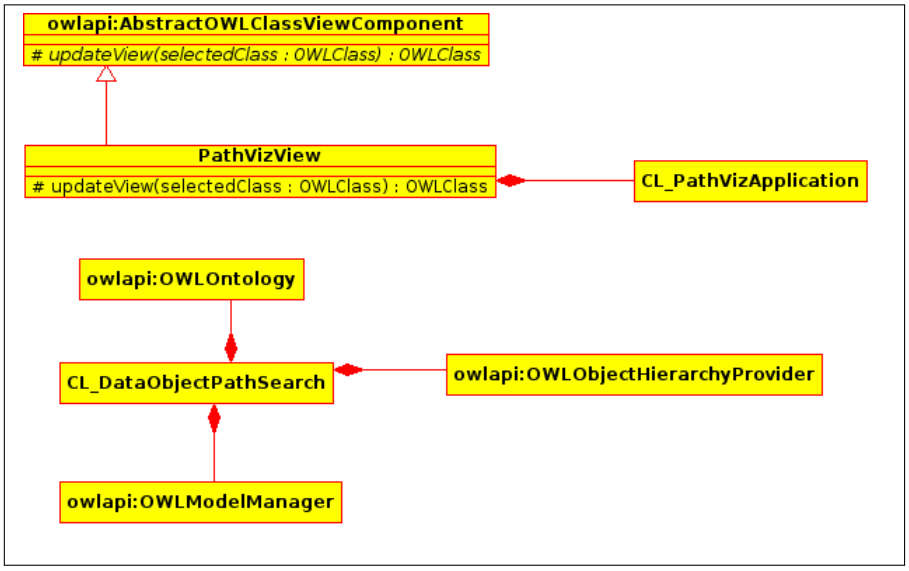


Figure B.3: PathViz and Protégé

notifier when something in the ontology changes. The class `PathVizView` contains the class `CL_PathVizApplication` that was described in the previous section.

The class `CL_DataObjectPathSearch` contains three auxilary classes that were obtained via the OWL API. With the class `OWLObjectHierarchyProvider` an asserted an inferred taxonomy of the ontology can be obtained. The classes `OWLOntology` and `OWLModelManager` are used to extract axioms and class relationships from the ontology.

C

Survey

C.1 Introduction

Respondents are requested to complete this survey, as part of an evaluation of PathViz as a software tool that aids ontology comprehension. Your honest response is highly valued, and will be treated with strict confidentiality. Please take care to answer the following questions thoughtfully, accurately and fairly. Give your own opinion, indicating whether you agree or disagree with each statement. Use the following key, and circle the item on the rating scale that corresponds best to your experience:

1 = Strongly disagree

2 = Disagree

3 = Neutral

4 = Agree

5 = Strongly agree

This survey is anonymous. Thank you for your participation.

C.2 Questions

Comprehension					
PathViz facilitates my understanding of an ontology	1	2	3	4	5
PathViz helps me to understand how two concepts in an ontology are related to each other	1	2	3	4	5
PathViz helps me to understand what effect the ontology reasoner has on the way concepts in the ontology relate to each other	1	2	3	4	5
1–Strongly disagree ... 5–Strongly agree					

Usability					
PathViz is easy to use	1	2	3	4	5
PathViz makes effective use of colours to highlight relevant information	1	2	3	4	5
PathViz is easy to navigate	1	2	3	4	5
1–Strongly disagree ... 5–Strongly agree					

Definitions					
An ontology comprehension framework is a useful mechanism to describe and categorise software tools that aid ontology comprehension	1	2	3	4	5
The *path cardinality ratio* is a useful measurement instrument for indicating the influence the ontology reasoner has on the way concepts in the ontology relate to each other	1	2	3	4	5
The *path simplicity ratio* is a useful measurement instrument for indicating the influence the ontology reasoner has on the way concepts in the ontology relate to each other	1	2	3	4	5
1–Strongly disagree ... 5–Strongly agree					

C.3 Other comments

Please take some time to comment on other positive or negative aspects relating to PathViz and ontology comprehension. All comments, ideas and suggestions are welcome.

Bibliography

[1] OWL 2 Web ontology language – Document overview. Retrieved on 2010/11/01 from http://www.w3.org/TR/owl2-overview/.

[2] OWLViz URL. Retrieved on 2010/01/01 from http://www.co-ode.org/downloads/owlviz/.

[3] Pizza URL. Retrieved on 2010/01/01 from http://owl.cs.manchester.ac.uk/2009/esslli-explanation/pizza.owl.

[4] Protégé developer docs (1). Retrieved on 2010/01/01 from http://protegewiki.stanford.edu/wiki/Protege4DevDocs.

[5] Protégé developer docs (2). Retrieved on 2010/01/01 from http://protegewiki.stanford.edu/wiki/P4APIOverview.

[6] Protégé developer docs (3). Retrieved on 2010/01/01 from http://protege.stanford.edu/protege/4.0/docs/api/.

[7] RSA URL. Retrieved on 2010/01/01 from http://www.afrilux.co.za/images/maps/south_africa_map.gif.

[8] SNOWMED URL. Retrieved on 2010/01/01 from http://www.ihtsdo.org/snomed-ct/.

[9] Tree URL. Retrieved on 2010/01/01 from http://en.wikipedia.org/wiki/File:BasicTree.png.

[10] VEN URL. Retrieved on 2010/01/01 from http://en.wikipedia.org/wiki/File:Venn_diagram_cmyk.svg.

[11] H. Alani. TGVizTab: An ontology visualization extension for Protégé. In *Knowledge capture (K-Cap'03) - Workshop on visualization information in knowledge engineering*, 2003. Retrieved on 2010/01/01 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.2433&rep=rep1&type=pdf.

[12] G. Antoniou and F. Harmelen. Web ontology language: OWL. *Handbook on ontologies*, pages 91–110, 2009.

[13] F. Baader. Appendix 1: Description logic terminology. *The Description logic handbook: Theory, implementation, and applications*, pages 485–495, 2003.

[14] F. Baader, I. Horrocks, and U. Sattler. *Handbook on ontologies*, chapter 1. Springer, 2004.

[15] F. Baader and W. Nutt. Basic description logics. In *The Description logic handbook*, pages 43–95. Cambridge University Press, 2003.

[16] J. Bauer. Model exploration to support understanding of ontologies. Master's thesis, Technische Universität Dresden, 2009. Retrieved on 2010/01/01 from `http://www.tatome.de/uni/d_thesis.pdf`.

[17] J. Bauer, U. Sattler, and B. Parsia. Explaining by example: Model exploration for ontology comprehension. 2009. Retrieved on 2010/01/01 from `http://ceur-ws.org/Vol-477/paper_37.pdf`.

[18] K. Börner, C. Chen, and K.W. Boyack. Visualizing knowledge domains. *Annual review of information science and technology*, 37(1):179–255, 2003.

[19] A. Bosca, D. Bonino, and P. Pellegrino. OntoSphere: More than a 3D ontology visualization tool. In *SWAP, the 2nd Italian semantic web workshop*, 2005. Retrieved on 2010/01/01 from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.8462&rep=rep1&type=pdf`.

[20] B.J. Calder, L.W. Phillips, and A.M. Tybout. The concept of external validity. *Journal of Consumer Research*, 9(3):240–244, 1982.

[21] V.K. Chaudhri, M.E. Stickel, J.F. Thomere, R.J. Waldinger, et al. Using prior knowledge: Problems and solutions. In *National conference on artificial intelligence*, pages 436–442, 2000.

[22] R. Cornet and A. Abu-Hanna. Usability of expressive description logics – a case study in UMLS. In *Proceedings of the AMIA symposium*, page 180. American Medical Informatics Association, 2002.

[23] C. Fluit, M. Sabou, and F. Van Harmelen. Supporting user tasks through visualisation of light-weight ontologies. *Handbook on ontologies*, pages 415–434, 2004.

[24] C. Fluit, M. Sabou, and F. Van Harmelen. Ontology-based information visualization: Toward semantic web applications. *Visualizing the semantic web*, pages 45–58, 2006.

[25] A. Gibson, K. Wolstencroft, and R. Stevens. Promotion of ontological comprehension: Exposing terms and metadata with web 2.0. In *Workshop on social and collaborative construction of structured knowledge at 16th International World Wide Web Conference*, 2007. Retrieved on 2010/01/01 from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.8462&rep=rep1&type=pdf`.

[26] B.C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Web semantics: Science, services and agents on the World Wide Web*, 6(4):309–322, 2008.

[27] T. Gruber. Towards principles for the design of ontologies for knowledge sharing. *International journal of human computer studies*, 43(5/6):907–928, 1995.

[28] N. Guarino. Formal ontology, conceptual analysis and knowledge representation. Retrieved on 2010/01/01 from `http://www.loa-cnr.it/Papers/FormOntKR.pdf`.

[29] N. Guarino. Formal ontology in information systems. In *FOIS'98*, pages 3–15. IOS Press, 1998.

[30] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools. Retrieved on 2010/01/01 from `http://www.geog.ucsb.edu/~raubal/Courses/288MR_Spring08_Papers/ProtegeOWLTutorial.pdf`, March 2009.

[31] I. Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51(12):58–67, 2008.

[32] I. Horrocks, P.F. Patel-Schneider, and F. Van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[33] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ (D) description logic. In *International joint conference on artificial intelligence*, volume 17, pages 199–204, 2001.

[34] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *6th International semantic web conference and 2nd Asian semantic web conference*, pages 267–280. Springer-Verlag, 2007.

[35] A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca-Grau. Repairing unsatisfiable concepts in owl ontologies. *The semantic web: Research and applications*, pages 170–184, 2006.

[36] A. Kalyanpur, B. Parsia, E. Sirin, B.C. Grau, and J. Hendler. Swoop: A web ontology editing browser. *Web semantics: Science, services and agents on the World Wide Web*, 4(2):144–153, 2006.

[37] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods – A survey. *ACM computing surveys*, 39(4):10, 2007.

[38] C. Keet. Enhancing comprehension of ontologies and conceptual models through abstractions. *Artificial intelligence and human-oriented computing*, pages 813–821, 2007.

[39] I. Niskanen, J. Kalaoja, J. Kantorovitch, and T. Piirainen. An interactive ontology visualization approach for the networked home environment. *International journal of computer and information science and engineering*, 1(2):102–107, 2007.

[40] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *14th International conference on World Wide Web*, pages 633–640. ACM, 2005.

[41] B. Parsia, T. Wang, and J. Golbeck. Visualizing web ontologies with cropcircles. In *End user semantic web interaction*, pages 6–10. ISWC, 2005.

[42] J. Preece, Y. Rogers, and H. Sharp. *Interaction design*. Wiley, 2002.

[43] A.J. Pretorius. Ontologies – Introduction and overview. *Semantic technology and applications research laboratory*, 2004. Retrieved on 2010/01/01 from `https://www.starlab.vub.ac.be/teaching/Ontologies_Intr_Overv.pdf`.

[44] A. Rosendahl. Visualization of knowledge in the eParticipation ontology. Retrieved on 2009/12/02 from `http://www.imu.iccs.gr/events/kmworkshop/index_files/05-Ontology-Visualization-final.doc`., 2006.

[45] R. Singleton, B.C. Straits, and M.M. Straits. *Approaches to social research*. Oxford University Press New York, 1993.

[46] R. Volz. Semantics at work: Ontology management – Tools and techniques. Retrieved on 2010/01/01 from `http://www.lulu.com/items/volume_62/1969000/1969742/1/print/SemanticsAtWork3.pdf`.

[47] H. Wang, M. Horridge, A. Rector, N. Drummond, and J. Seidenberg. Debugging OWL-DL ontologies: A heuristic approach. *The semantic web–ISWC*, pages 745–757, 2005.

[48] C. Ware. *Information visualization*. Morgan Kaufmann, 2000.